VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Tomáš Vostřel

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍDICÍ INTEGROVANÝ SYSTÉM PRO ROZPOZNÁVÁNÍ OBROBKŮ
CONTROL INTEGRATED SYSTEM FOR WORKPIECE RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tomáš Vostřel**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Petr Pivoňka, CSc.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Tomáš Vostřel ***ID:*** 186234
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Řídicí integrovaný systém pro rozpoznávání obrobků

POKYNY PRO VYPRACOVÁNÍ:

1. Proveďte rešerši použití strojového vidění v průmyslu.
2. Seznamte se s řídicími prostředky společnosti B&R Automation a vývojovým prostředím Automation Studio.
3. Ověřte možnosti použití inteligentní kamery pro rozpoznávání a určení umístění obrobků a navrhněte koncepci softwarového řešení.
4. Implementujte navržené řešení do PLC.
5. Doplňte vizualizaci na dotykovém panelu.
6. Ověřte funkčnost realizace dostupnými prostředky.

DOPORUČENÁ LITERATURA:

[1] Firemní literatura B&R: Automation Studio V4

[2] Firemní literatura B&R: Produkty pro řízení pohybu a vizualizace společnosti B&R

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: prof. Ing. Petr Pivoňka, CSc.
Konzultant: Ing. Růžička Kamil, ABB B+R Brno

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zaobírá použitím integrovaného strojového vidění firmy B&R, tzv. *Smart Sensoru*, k rozpoznávání a určení pozice kovových obrobků obdélníkového tvaru. V rámci práce je provedena rešerše použití strojového vidění v průmyslu, navrženo a realizováno programové řešení a vytvořena uživatelská vizualizace. Hlavním výsledkem je knihovna VITemplate, jejíž použití v kombinaci s Blob analýzou založenou na modelech implementovanou v kameře umožňuje řídit robotické rameno tak, aby mohly být obrobky úspěšně odebrány z pásu.

KLÍČOVÁ SLOVA

strojové vidění, rozpoznávání obrobků, B&R, Smart Sensor, Blob analýza, B&R Automation, Automation Studio

ABSTRACT

The diploma thesis deals with the usage of integrated machine vision by B&R, *Smart Sensor*, for metal rectangular-shaped workpiece recognition and position determinition. The description of the usage of machine vision in the industry is made, the solution concept is created and the program and the user visualisation implemented. The main outcome of this work is the VITemplate library which can be used in combination with the model-based Blob analysis implemented in Smart Sensor to control the robotic arm to successfully grab all the workpieces on the belt.

KEYWORDS

machine vision, workpiece recognition, B&R, Smart Sensor, Blob analysis, B&R Automation, Automation Studio

VOSTŘEL, Tomáš. *Řídicí integrovaný systém pro rozpoznávání obrobků*. Brno, 2020, 58 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: prof. Ing. Petr Pivoňka, CSc.

PROHLÁŠENÍ

Prohlašuji, že svůj semestrální projekt na téma „Řídicí integrovaný systém pro rozpoznávání obrobků“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 25.5.2020 .....

podpis autora

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu diplomové práce panu prof. Ing. Petru Pivoňkovi, CSc. za vstřícné jednání, odborné vedení a podnětné připomínky k práci.

Brno 25.5.2020 .....

podpis autora

B+R automatizace, spol. s r.o.
Stránského 39
616 00 Brno
Česká republika
https://www.br-automation.com/

PODĚKOVÁNÍ

Předmět této diplomové práce byl realizován v brněnské pobočce společnosti B&R. Rád bych poděkoval za vstřícné jednání, se kterým jsem se setkal, a to jmenovitě především konzultantovi práce Ing. Kamilu Růžičkovi, dále také Ing. Tomáši Matuchovi, Ph.D. a Ing. Zdeňku Švihálkovi.

Brno 25.5.2020 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Strojové vidění (*Machine Vision, MV*) zaujímá v posledních letech v rámci průmyslových systémů stále pevnější postavení. Nasazením prostředků MV lze dosáhnout realizace úloh, které by dříve realizovat možné nebylo, nebo jen s obtížemi. V roce 2020 vstupuje společnost B&R na trh se Smart Sensorem, první průmyslovou kamerou, která je inteligentní a plně integrovaná do řídicího systému. [1]

Jednou ze základních průmyslových úloh řešitelných za pomoci prostředků MV je úloha rozpoznávání objektů za účelem řízení robota. Rozpoznáváním objektů je v případě této práce zamýšlen proces, který vede k určení umístění, orientace a rozměrů (délky a šířky) kovových obrobků přijíždějících pod kameru na páse. Zadání je zjednodušeno několika výchozími předpoklady; všechny obrobky jsou tvaru kvádru známé výšky, jsou ze surového nebo opracovaného železa, pod Smart Sensor najíždí na páse, jehož barvu je možné do jisté míry specifikovat a jež se během procesu rozpoznávání nepohybuje, obrobky jsou na páse rozmístěny v jedné vrstvě a vzájemně se nedotýkají. Požadavkem je obrobky odebírat robotickým ramenem.

Popsaná úloha je základním tématem této diplomové práce a vyvstala během mé studentské praxe v brněnské pobočce B&R, při které mi bylo umožněno se v průběhu roku 2019 seznámit a pracovat s prototypem Smart Sensoru ještě před jeho oficiálním uvedením na trh. Smart Sensor nativně možnost nasazení při řízení robota nenabízí, zdá se však, že by k řešení nastíněné úlohy použit být mohl, protože jsou v něm implementovány dva algoritmy sloužíci k rozpoznávání objektů v obraze. Diplomová práce si klade za cíl ověřit tuto možnost a případně navrhnout koncepci řešení a provést jeho realizaci.

1 Strojové vidění v průmyslových aplikacích

V krátkém úvodu do široké oblasti strojového vidění jsou diskutovány specifika systémů strojového vidění určených pro průmyslové aplikace a provedena jejich rešerše. V neposlední řadě je zde věnována pozornost základním postupům při zpracování obrazu pro identifikaci objektů.

1.1 Pojem strojového vidění

Strojové vidění (*Machine Vision, MV*) hraje v současných výrobních procesech roli, jejíž důležitost roste spolu se zvyšujícími se nároky na automatizaci výroby, kvalitu výrobků a jejich sledovatelnost napříč výrobním cyklem. [2]

Pojem strojového vidění není snadné přesně definovat, většina pokusů o to se tedy více či méně liší. Nicméně většina z nich do strojového vidění zahrnuje všechny prostředky a metody, jakými lze ze snímku získat informace využitelné v navazujících aplikacích. Tyto aplikace/úkoly jsou definovány např. v publikaci „Guideline for Industrial Image Processing“ [3], a to následujícím způsobem.

- Identifikace nebo rozpoznávání objektu. Identifikace probíhá na základě speciálních identifikačních symbolů, např. čárových nebo QR kódů, rozpoznávání na základě charakteristických vlastností objektu, např. tvaru nebo rozměrů.
- Detekce pozice objektu. Může být prováděna ve dvou nebo trojrozměrném prostoru, v závislosti na aplikaci, kterou může být například navigace robota.
- Kontrola úplnosti objektu. Ta je typicky prováděna po ukončení určité části výrobního procesu, před zahájením další.
- Zjišťování geometrických vlastností. Zde se jedná především o tvar a rozměry objektu. Tyto údaje mohou posloužit jednak jako měřítko kvality, jednak jako důležitá informace pro úchop robotem.
- Kontrola povrchu objektu a hledání nedostatků (kontrola kvality zpracování).

Počítačové vs. strojové vidění

Počítačové vidění (*Computer Vision, CV*) a strojové vidění jsou dva rozdílné termíny pro překrývající se obory. Počítačové vidění se zabývá pořízením a analýzou obrazové informace z obecnějšího, spíše vědeckého hlediska. Důraz klade především na způsob analýzy obrazu – a to napříč velkou škálou teoretických a praktických aplikací. Pojem strojové vidění se tradičně používá spíše pro nasazení počítačového vidění v konkrétních průmyslových a praktických aplikacích. Jedná se tedy spíše o inženýrskou disciplínu, ve které je důraz kladen, spíše než na samotnou analýzu obrazu, na to, jak jí využít dále – pro řízení výrobního procesu, navigaci robota,

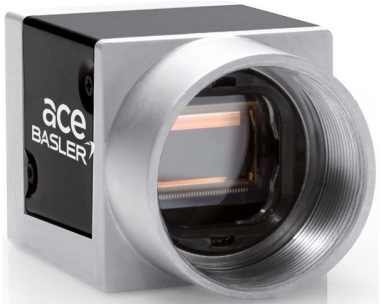
apod. [4], [5], [6] Různé zdroje však pojmy počítačového a strojového vidění definují odlišně.

1.2 Integrace strojového vidění do průmyslu

V zásadě lze z hlediska HW rozlišovat následující stupně integrace MV (strojového vidění) do řídicích systémů průmyslových aplikací.

1.2.1 Bez integrace strojového vidění

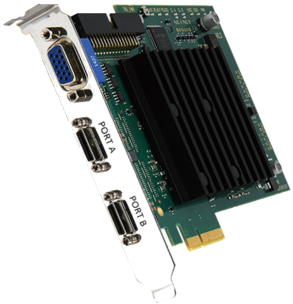
Hlavními komponentami systému strojového vidění jsou kamera samotná (myšleno obrazový snímač umístěný v těle kamery), optická soustava (objektivy, mezikroužky, filtry apod.), prvky osvětlení, frame grabber a algoritmy zpracování obrazu. [6] Všechny tyto komponenty jsou na současném trhu dobře zastoupeny. Z HW hlediska mají velkou nabídku například následující výrobci: Cognex, Teledyne, Basler, Datalogic, Edmund Optics, Ricoh, FLIR, Adimec, PixeLINK, Manta, Balluff atd. Z hlediska SW lze využít open-source řešení např. v podobě knihovny OpenCV nebo SOD Embedded, zakoupit komerční řešení (např. nástroj Cognex VisionPro nebo knihovnu HALCON společnosti MVTec Software GmbH) nebo samozřejmě vyvíjet řešení vlastní.



Obr. 1.1: Kamera Basler ace classic [7]

Obr. 1.2: Objektivy Basler Lenses 2/3" [8]

Obr. 1.3: Basler osvětlení [9]



Obr. 1.4: Frame gr. microEnable 5 [10]

1.2.2 Integrované strojové vidění

Někteří výrobci se v posledních letech začali zabývat vývojem tzv. „chytrých kamer“ (anglicky *smart sensor*, případně *smart camera*), což jsou zařízení, která již mají určitou míru integrace strojového vidění danou z výroby. Tato integrace spočívá v tom, že obsahují vlastní procesor a předem připravené základní algoritmy pro předzpracování a analýzu obrazu. Nejčastěji jsou to algoritmy pro čtení čárových a QR kódů, rozpoznání textu (*Optical character recognition, OCR*), měření nebo rozpoznávání tvarů a vzorů. Velmi často disponují vestavěným osvětlením a jejich použití je velmi snadné a uživatelsky pohodlné. Příkladem takové chytré kamery může být například model NEON-1021-M od firmy Adlink, Sharpshooter od společnosti Balluff, Cognex In-Sight 7000, Leuze LSIS 400i Series nebo právě Smart Sensor od B&R – ten je však oproti předchozím co se týče integrace ještě o úroveň výše, neboť je plně integrován i do řídicího systému (více v následující části 1.2.3).

1.2.3 Integrované strojové vidění součástí řídicího systému

Významnosti strojového vidění v průmyslových aplikacích si v posledních letech začali všímat též výrobci řídicích systémů a přichází s prvními řešeními. Na současném trhu jsem našel dva výrobce řídicích systémů, kteří vyvíjí své chytré kamery; je to B&R Automation a Omron Industrial Automation. Základní popis systémů těchto výrobců bude proveden dále. V minulosti vyvíjela chytrou kameru též společnost Rockwell Automation (Allen-Bradley), jejich 48MS MultiSight sensor již však není dostupný, proto se jím zabývat nebudu. [11] Jinou cestou se dává firma Beckhoff, která strojové vidění integruje pouze po SW stránce a HW umožňuje zvolit libovolný s rozhraním GigE Vision. [12]

B&R Smart Sensor, SmartCamera

B&R uvádí na trh dvě varianty inteligentní kamery: Smart Sensor a Smart Camera. Smart Sensor byl v době vzniku této práce v pilotní, prototypální fázi. Na jaře roku 2020 bude Smart Sensor běžně dostupný, Smart Camera započne svou pilotní fázi. Oproti Smart Sensoru se bude lišit možností simultánního běhu více Vision funkcí.

Ve snaze pokrýt široké spektrum aplikací strojového vidění nabízí B&R více možností ve volbě obrazových snímačů, objektivů, osvětlení a dalšího příslušenství. Po HW stránce je zajímavý elektronický pohon ostření objektivu, FPGA čip pro předzpracování obrazu nebo vestavěné osvětlení v podobě čtyř segmentů po čtyřech výkonných LED diodách emitujících buď červené, zelené, modré, zeleno-žluté, bílé světlo, UV nebo IR záření. Kamera je připojena jediným kabelem, který sdružuje napájení 24 V a komunikaci přes Powerlink. Kamery lze řetězit za sebe, stejně tak

přídavné externí osvětlení. Oba modely jsou od výroby softwarově vybaveny následujícími pěti Vision funkcemi: Blob analýzou založenou na modelech, Matchingem, čtením čárových a QR kódů, měřením a OCR. V případě zájmu o další informace odkazuji na internetové stránky výrobce. [1]

Obr. 1.5: B&R Smart Sensor [1]

Obr. 1.6: Omron FHV7 Series [13]

Omron Smart Camera FHV7 Series

Společnost Omron nabízí tři různé modely chytrých kamer. Model FQ-M je navržený speciálně pro Pick & Place aplikace, pro navigaci robotů a sledování pohybu, model FQ2 je univerzálnější a lze jej vnímat jako předchůdce modelu FHV7, který je vhodný pro většinu běžných aplikací strojového vidění.

Omron FHV7 je v mnoha ohledech podobný již popsanému Smart Sensoru od B&R. Taktéž lze u něho vybírat z velkého množství konfigurací s různými obrazovými snímači, objektivy, osvětlením a dalším příslušenstvím. Oproti Smart Sensoru lze volit barevný snímač obrazu, větší rozlišení čipu (až 12 Mpix), objektiv s tekutou čočkou (*liquid lens*) a krytí IP67. Ze SW hlediska je Omron FHV7 vybaven funkcí pro čtení čárových kódů, OCR, funkcí Shape Search pro vyhledávání objektů, nástroji pro předzpracování obrazu, kontrolu, měření a spolupráci s roboty různých výrobců. Pro další informace doporučuji nahlédnout do katalogu výrobce dostupného z jeho internetových stránek. [13]

1.3 Zpracování obrazu pro identifikaci objektů

Zpracováním obrazu za účelem identifikace objektu se v posledních 50 letech zabývalo a stále zabývá velké množství odborníků, je o něm zveřejněno množství publikací a stále jsou nacházeny nové metody a přístupy. Popsat i jen ty nejpoužívanější by bylo vysoko nad rámec této práce. Nezbývá mi tedy, než případné zájemce odkázat

na dostupnou literaturu a zabývat se zde pouze algoritmy implementovanými ve Smart Sensoru, který budu v práci dále používat.

Tyto algoritmy mají podobu Vision funkcí a jsou přítomny v komponentě *mapp Vision*. Vision funkce pro čtení kódů, měření a OCR se k zpracování obrazu pro identifikaci objektů nehodí. V úvahu připadají následující dvě: Blob analýza založená na modelech nebo Matching.

1.3.1 Blob analýza založená na modelech (*Model-based Blob Analysis*)

Blob analýza umožňuje nalézt objekty na základě jejich jasových hodnot v obraze. Základním předpokladem pro její použití je dostatečný kontrast mezi hledanými objekty a pozadím. Typicky toho lze dosáhnout umístěním tmavých objektů na světlý podklad nebo opačně. Pak mají jasové hodnoty, jichž objekt na snímku nabývá, dostatečný odstup od jasových hodnot, jichž nabývá pozadí. Výsledkem analýzy jsou souvislé oblasti, tzv. *blobs*, které mají jasové hodnoty z intervalu náležícímu objektu.

Jaké jasové hodnoty náleží hledanému objektu stanovuje jeho model. Modely se vytváří pomocí nástroje nazvaného *Vision Cockpit*. Jedná se o HMI server bežící v řídicím kontroleru (PLC nebo APC), ke kterému se lze připojit standartním internetovým prohlížečem z počítače na stejné TCP/IP síti. Výměna dat mezi HMI serverem a kamerou je zajištěna propojením skrze rozhraní Powerlink. Při vytváření modelu je hledaný objekt popsán krom jasových hodnot i dalšími vlastnostmi, které lze vyjádřit číslem, tzv. příznaky. Těmito příznaky jsou plocha, obdélníkovost, kruhovost a anisometrie. Jakých hodnot příznaků může objekt nabývat je stanoveno intervaly (*Min* až *Max*).

- **ThresholdMin** – **Max**: jasové hodnoty, jichž objekt nabývá.
- **AreaMin** – **Max**: počet pixelů, které tvoří plochu objektu.
- **RectangularityMin** – **Max**: obdélníkovost objektu (podobnost obdélníku). Pro její stanovení je nejprve vypočten obélník, který má moment prvního a druhého řádu stejný jako objekt. Výpočet obdélníkovosti je podle metody Paul L. Rosina [14] následující:

$$Rectangularity = 1 - \frac{R + D}{RectangleArea}\ [-], \tag{1.1}$$

  kde R je rozdíl ploch obdélníku a objektu, D rozdíl ploch objektu a obdélníku a *RectangleArea* plocha obdélníku. Může nabývat hodnot 0 až 1.
- **CircularityMin** – **Max**: kruhovost (podobnost kruhu) objektu definovaná:

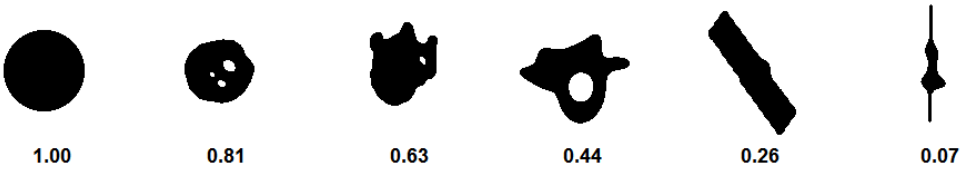
$$Circularity = min\left(1, \frac{Area}{MaxDist^2 \cdot \pi}\right)\ [-], \tag{1.2}$$

kde *Area* je plocha objektu a *MaxDist* je maximální vzdálenost středu od pixelů tvořících hranici objektu. [15] Nabývat může opět hodnot z intervalu 0 až 1.

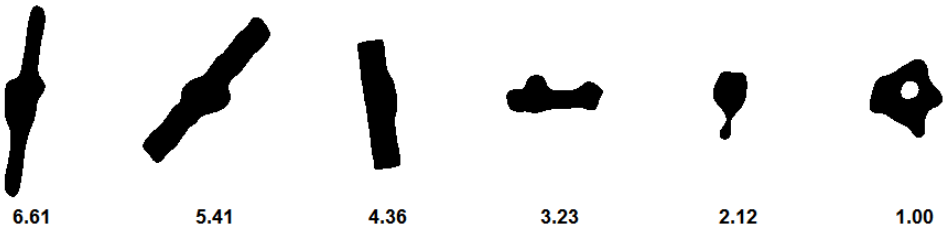
- **AnisometryMin − Max**: anisometrie objektu. Pro stanovení anisometrie jsou nejprve zjištěny délky os elipsy, která má stejnou orientaci a poměr stran jako objekt. Délka hlavní osy je značena R_a, délka vedlejší R_b. Anisometrie objektu je pak spočtena následovně ([15]):

$$Anisometry = \frac{R_a}{R_b}\ [-]. \tag{1.3}$$

Obr. 1.7: Hodnoty obdélníkovosti pro různé tvary objektu [15]



Obr. 1.8: Hodnoty kruhovosti pro různé tvary objektu [15]



Obr. 1.9: Hodnoty anisometrie pro různé tvary objektu [15]

Dalším parametrem specifickým pro každý model je volba *MorphologyType*, kterou lze vybrat, zda se má při hledání objektu provádět morfologická operace uzavření (*MorphologyType = 1*) nebo otevření (*MorphologyType = 2*). V případě použití morfologických operací je též nutno definovat velikost nástroje *CircleMaskRadius*.

Blob analýza vrací pro každý nalezený objekt mající příznaky některého z modelů následující: identifikační číslo modelu, zda je objekt oříznut (okraj snímku), jaká je plocha objektu, pozice ve směru osy X a Y, délka, šířka, orientace a střední jasová úroveň.

1.3.2 Matching

Matching funkci lze použít k vyhledávání objektů a určení jejich pozice a orientace pomocí modelů. Z tohoto hlediska se jedná o nástroj velmi podobný Blob analýze, funguje ale jiným způsobem a je vhodný pro jiné aplikace. Ve Smart Sensoru jsou implementovány dvě varianty Matching funkce, z nichž je každá postavena na jiném principu. První z nich identifikuje objekt v obraze pomocí NCC; normalizované vzájemné korelace (*normalized cross-correlation*). Druhá varianta pak provádí identifikaci na základě modelu tvaru objektu (*shape-based*).

Obě varianty mají společné parametry *NumSearchMax* (maximální počet nalezených objektů), *Timeout* (časový limit v ms), *MinScore* (minimální shoda objektu s modelem v %), *MaxOverlap* (maximální povolené překrytí objektu v %), *OffsetROIX*, *OffsetROIY* a *OffsetROIOrientation* (stanovení posunu a orientace ROI – oblasti zájmu (*region of interest*)). Společnými výstupy obou funkcí jsou identifikační číslo modelu, pozice nalezeného objektu (v 1/100 px), jeho orientace, relativní velikost vůči modelu *Scale* a procentuální shoda s modelem *Score*.

Matching založený na NCC

Jako model je při použití NCC použit přímo výřez ze snímku. Při hledání objektu v obraze je počítána podobnost modelu s částí obrazu pro všechna přípustná posunutí modelu po obrazu. Výpočet NCC pro posunutí (x', y') modelu $m(x, y)$ v obraze $f(x, y)$ je dán vztahem:

$$NCC(x', y') = \frac{1}{N} \sum_{(x,y) \in ROI} \frac{m(x, y) - \overline{m}}{\sqrt{s_m^2}} \cdot \frac{f(x' + x, y' + y) - \overline{f}(x', y')}{\sqrt{s_f^2(x', y')}} \; [-], \quad (1.4)$$

kde N je počet pixelů modelu, ROI oblast modelu, $\overline{m}$ průměrná jasová úroveň modelu, s_m^2 střední kvadratická odchylka jasové úrovně modelu, $\overline{f}(x', y')$ průměrná jasová úroveň a $s_f^2(x', y')$ střední kvadratická odchylka jasové úrovně obrazu pro posun (x', y') v oblasti dané ROI. [15]

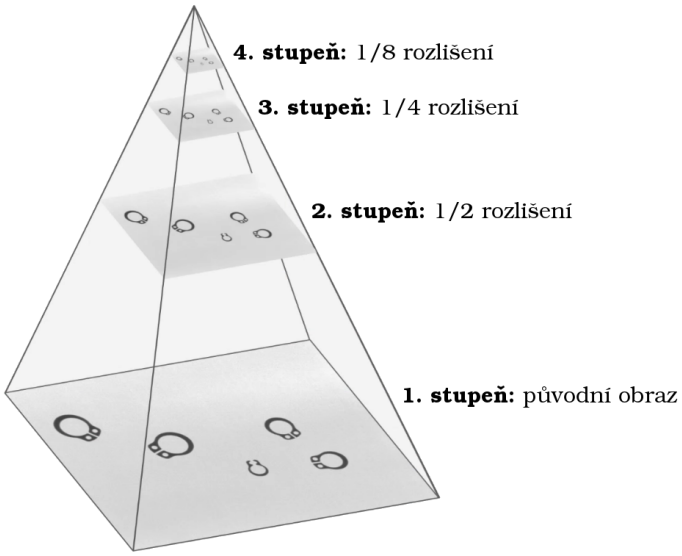
Získané výsledky NCC leží mezi -1 a 1, čím větší je absolutní hodnota výsledku, tím větší je míra korespondence mezi modelem a výřezem obrazu. Hodnota 1 znamená, že jasové hodnoty výřezu obrazu jsou lineární transformací jasových hodnot modelu. Vůči těmto jasovým změnám je tedy metoda Matchingu založená na NCC invariantní.

Při tvorbě modelu lze použít parametr *NCCSearchSubPixels*, který volí vyhledávání objektu se subpixelovou přesností. Oproti druhé variantě popsané následně je Matching založený na NCC výhodnější při hledání objektů, jejichž tvar není vždy přesně stejný, jež mají výrazně strukturovaný povrch nebo jsou na snímku zachyceny rozmazaně. Bývá však, obzvláště při větších rozměrech modelu, časově náročnější.

Shape-based Matching

Jak napovídá název této metody, hledané objekty jsou v obraze popsány a rozpoznány na základě jejich tvaru. Způsobů, jak zjistit a popsat tvar objektu, je samozřejmě mnoho. Zde je tvar popsán pixely, u nichž dochází k jasovým změnám vůči okolí větším, než je stanovený práh – povětšinou tyto body odpovídají obrysu hledaného objektu. Při vytváření modelu jsou specifikovány následující parametry, některé z nich jsou duplikovány a lze je použít zvlášť při tvorbě modelu a při hledání objektů. Při jejich popisu jsem vycházel z publikace HALCON Solution Guide II-B Shape-based Matching. [16]

- **ShapeSearchGreediness**: parametr, kterým lze ovlivnit samotný algoritmus hledání objektu. Vyvažuje se jím důkladnost a rychlost algoritmu. Při nastavení hodnoty 0 je kladen důraz na důkladnost hledání; je tedy časově náročné. Naopak nastavením hodnoty 1 je více prioritizována rychlost; je tedy větší pravděpodobnost nenalezení objektu.
- **ShapeSearchBorderShapeModels**: pokud není tímto parametrem stanoveno jinak, nejsou objekty, které jsou zčásti mimo snímek, hledány.
- **ModelNumLevels**: počet stupňů pyramidy, kterou je možno pro zrychlení hledání modelu v obraze vytvořit z původního obrazu (základna pyramidy) a jeho podvzorkovaných variant. Každý z vyšších stupňů je tvořen obrazem složeným z poloviční množství pixelů oproti tomu předchozímu. Model je hledán prvně na nejvyšším stupni pyramidy (v nejvíce podvzorkovaném, tedy nejmenším obraze). Výsledek hledání je použit pro zmenšení oblasti, ve které je posléze model hledán na nižším stupni. Toto je opakováno, dokud není dosaženo nejnižšího stupně – původního obrazu.



Obr. 1.10: Znázornění pyramidy vytvořené podvzorkováváním obrazu [17]

- **ModelAngleStart, -Extent, -Step**: parametry sloužící k definici úhlů, do kterých může být objekt v obraze pootočen (relativně k orientaci, při které byl tvořen model) a stále být nalezen. Omezením možné orientace objektu dochází ke zrychlení procesu hledání. Další důvod k omezení možné orientace objektu vyvstává v okamžiku, kdy je objekt (téměř) symetrický. Aby nebylo nalezeno více modelů na stejném místě s různou orientací, je omezení nutné. Pro objekt tvaru kříže nebo čtverce je maximální rozsah menší než 90°, pro obdélník menší než 180, pro kruhový objekt 0°. Pro automatické nastavení těchto parametrů lze zvolit možnost *auto*.
- **ModelMetric:** parametr specifikující, zda je při hledání pixelů stanovujících tvar objektu nahlíženo k polaritě (směru) jasové změny. Volba „use_polarity“ říká, že jasové změny všech těchto pixelů musí mít stejnou polaritu (tedy např. světlý objekt na tmavém pozadí nebo naopak). Když je parametr ModelMetric roven možnosti „ignore_global_polarity“, je možné, aby se polarita měnila (světlý objekt na tmavém pozadí i naopak), zvýší se však časová náročnost. Při výběru „ignore_local_polarity“ může docházet i k lokálním změnám polarity (některé části objektu jsou oproti pozadí tmavé, jiné světlé) – tato možnost vede k velkému zvýšení časové náročnosti, obzvlášť v kombinaci s velkým rozsahem možných orientací (viz předchozí odrážka).
- **ShapeModelScaleMin, -Max, -Step**: parametry sloužící k definici velikostí objektu (relativně vzhledem k modelu), kterých může nabývat a stále být nalezen. Velikost je zadávána jako násobek velikosti modelu. Podobně jako tomu bylo dříve lze zvýšit rychlost volbou co nejmenšího rozsahu velikostí.
- **ShapeModelOptimization**: optimalizace modelu. Standartně jsou ukládány všechny body tvořící model objektu, pokud je jich ale velké množství, lze jejich počet vhodnou volbou parametru ShapeModelOptimization redukovat: „point_reduction_low, point_reduction_medium, point_reduction_high“ nebo „auto“.
- **ShapeSearchSubPixels**: parametr, který volí vyhledávání objektu se subpixelovou přesností.

Funkce je použitelná i pro objekty, které jsou oproti modelu v jiné orientaci nebo mají jinou velikost. Mohou být též zkoseny nebo jinak perspektivně deformovány, zčásti zakryty nebo mimo snímek. Do značné míry jsou též nezávislé na nelineárních fluktuacích osvětlení. [1]

2 Kocepce řešení

Kapitola 2 je zaměřena na návrh koncepce řešení. Jsou v ní postupně rozebrány jednotlivé prvky, z nichž se řešení skládá.

2.1 Popis scény a konfigurace Smart Sensoru

V této části je popsána a navržena scéna snímání. Zároveň s tím je posuzována vhodnost použití Smart Sensoru v dostupné konfiguraci: černobílý snímač Python 1300 s rozlišením 1280x1024 px a velikostí pixelu 4,8 μm, objektiv s ohniskovou vzdáleností 8 mm a clonovým číslem 4, vestavěné osvětlení s možným výběrem barvy světla. Při tvorbě scény bylo mé rozhodování značně omezeno již existujícím návrhem pracoviště, v jehož rámci bylo pevně stanoveno jak umístění kamery, tak dopravníku a robota. Kamera má být umístěna ve vzdálenosti přibližně 120 centimetrů nad pásem dopravníku. Pás je 80 cm široký a kamera by měla zabírat plnou šířku a alespoň 50 cm délky tohoto pásu.

Maximální úhlový zorný úhel (*angular field of view, AFOV*) je definován takto:

$$AFOV = 2 \cdot arctan\left(\frac{h}{2f}\right) \ [^\circ], \tag{2.1}$$

kde h je velikost čipu (mm) a f ohnisková vzdálenost (mm). [18] Maximální velikost zorného pole (*field of view, FOV*) v dané vzdálenosti od kamery $Dist$ (mm) lze podle stejného zdroje vypočítat následovně:

$$FOV = 2 \cdot Dist \cdot tan\left(\frac{AFOV}{2}\right) \ [mm]. \tag{2.2}$$

Zorné úhly Smart Sensoru v horizontálním a vertikálním směru lze získat dosazením do rovnice 2.1 následovně:

$$AFOV_{horizontal} = 2 \cdot arctan\left(\frac{4,8 \cdot 10^{-6} \cdot 1280}{2 \cdot 8 \cdot 10^{-3}}\right) \doteq 42,01 \ ^\circ, \tag{2.3}$$

$$AFOV_{vertical} = 2 \cdot arctan\left(\frac{4,8 \cdot 10^{-6} \cdot 1024}{2 \cdot 8 \cdot 10^{-3}}\right) \doteq 34,15 \ ^\circ. \tag{2.4}$$

Ve výšce 1200 mm nad pásem je maximální zorné pole kamery podle rovnice 2.2 následující:

$$FOV_{horizontal} = 2 \cdot 1200 \cdot tan\left(\frac{42,01}{2}\right) \doteq 921,60 \ [mm], \tag{2.5}$$

$$FOV_{vertical} = 2 \cdot 1200 \cdot tan\left(\frac{34,15}{2}\right) \doteq 737,28 \ [mm]. \tag{2.6}$$

Tato velikost zorného pole platí pro zaostření objektivu do nekonečna. Zaostření bude v dané aplikaci pravděpodobně jiné a zorné pole se o něco zmenší, i tak však lze usoudit, že z hlediska velikosti zorného pole je Smart Sensor v dostupné konfiguraci vyhovující.

Původně zvolená černá barva pásu nebyla shledána vhodnou z následujícího důvodu. Rozpoznávané obrobky mohou být ze surového železa nebo již opracované. Surové železo má spíše tmavou barvu, často na něm bývá patrná koroze a povrchové nečistoty. Opracované obrobky jsou oproti tomu velmi lesklé. Matný černý pás je vhodný pro rozpoznávání opracovaných lesklých obrobků, které se na jeho pozadí zdají světlé. Problém však nastává s tmavým neopracovaným železem, to s černým pásem na černobílých fotografiích Smart Sensoru poměrně značně splývá. Po konzultaci s dodavatelem dopravníku byla zvolena modrá barva pásu. Při nasvícení modrým světlem se pás jeví světlý a tmavé neopracované obrobky dobře vyniknou. Pro určení pozice lesklých opracovaných obrobků je pak použito červené světlo, ve kterém se modrý pás jeví tmavým a obrobky světlými. Z provedených experimentů se zdají být vestavěné LED diody pro osvícení scény dostačující.

K zaostření a nastavení expoziční doby bude použit automatický ostřící mechanismus Smart Sensoru. Protože je umístění kamery a pásu statické, mohou být získané hodnoty uloženy a Smart Sensor po restartu systému nastaven podle nich.

Na rychlost pořizování snímků nejsou kladeny vysoké nároky, pás není během snímaní v pohybu. Rychlost zpracování snímku též není kritická – obrábění je, stejně jako pohyb robota, relativně pomalé.

2.2 Volba Vision funkce

Vlastní zpracování pořízeného snímku za účelem získání pozice a rozměrů obrobků probíhá ve Smart Sensoru, přičemž je v první řadě nutné zvolit vhodnou Vision funkci, naparametrovat ji a případně též vytvořit modely. Začnu tedy výběrem Vision funkce. Jak bylo zmíněno v předchozím textu, v potaz přichází Blob analýza nebo některá ze dvou variant Matchingu. Postupovat budu vylučovací metodou.

Nejméně vhodným kandidátem je Matching založený na NCC. Tuto variantu Matchingu lze sice využít k určení pozice objektů, avšak spíše objektů složitějších, než jakými jsou obdélníkové obrobky. Za NCC model slouží přímo výřez ze snímku, což s sebou přináší nutnost vytvořit model pro každý obrobek. To není prakticky proveditlené, neboť obrobky mohou nabývat všech možných rozměrů v rozmezí 10 až 200 mm. Dalším důvodem, proč tuto metodu nelze v dané úloze použít, je fakt, že neposkytuje informaci o rozměrech nalezeného obrobku. Nevýhodou je též její časová náročnost.

Z velmi podobných důvodů lze vyloučit i druhou variantu Matchingu. Shape-based Matching pro svou funkci opět potřebuje modely obrobků konkrétních velikostí. Nutnost vytvořit model pro každý více či méně se lišící obrobek je velmi limitující. O velikosti nalezeného obrobku pak lze usuzovat pouze podle hodnoty *Scale*, což je relativní velikost vůči vytvořenému modelu. Nedostatečné rozlišení tohoto údaje a nutnost znalosti velikosti obrobku, který posloužil při tvorbě modelu, opět znemožňuje praktické nasazení v této aplikaci.

Poslední je Blob analýza. Její princip je sice velmi jednoduchý a nese s sebou nutnost dodržet dostatečný odstup jasových hodnot objektů od pozadí, lze však s výhodou využít principu tvorby modelů. Popis příznaky popsanými v části 1.3.1 je pro obdélníkové obrobky postačující a vhodné. Jediným modelem lze popsat obrobky i velmi různých rozměrů, protože jejich obdélníkovost a ostatní příznaky budou mít podobné hodnoty. Výhodou Blob analýzy je také fakt, že vrací rozměry nalezených obrobků se subpixelovou přesností (rozlišení 1/100 px). Oproti oběma variantám Matchingu lze též dosáhnout rychlejšího zpracování snímku.

Bylo provedeno několik experimentů, které výše uvedené úsudky o jednotlivých metodách potvrzují – za nejvhodnějšího kandidáta pro nasazení v reálné aplikaci opravdu vychází Blob analýza. Pokud je tvorbě modelu věnována dostatečná pozornost a je při ní použito větší množství různých obrobků různých velikostí, vykazuje Blob analýza vysokou úspěšnost nalezení. Přesná čísla o její spolehlivosti budou uvedena po odzkoušení na cílovém pracovišti. Při požadavku na hledání neopracovaných i opracovaných obrobků je nutné vytvořit modely pro obě možnosti a pořizovat dva snímky s různým nasvícením scény. Výsledky analýzy obou snímků pak mohou sloučeny dohromady. K problémům s nalezením dochází u obrobků s výraznými povrchovými vadami – metoda není vhodná například pro obrobky zpola opracované nebo velmi poškrábané. Koroze na povrchu surového železa nemá na nalezení obrobku významný vliv.

2.3 Zpracování dat získaných Blob analýzou

Data, která jsou získána Blob analýzou, nelze přímo použít pro navigaci robota; a to z několika důvodů. Pozice i rozměry nalezených objektů jsou udávány v pixelech, na snímcích se projevuje zkreslení dané perspektivou snímání a není známo, zda je okolo objektů dostatečný prostor pro chapadlo robota.

2.3.1 Kalibrace kamery

První operací, kterou je nad výsledky Blob analýzy nutné provést, je převod pixelů na milimetry. Tento proces bývá v literatuře označován jako kalibrace kamery a je

častou součástí systémů strojového vidění. V tomto případě je rovina snímání kolmá na osu kamery, díky čemuž se kalibrace pro danou vzdálenost od kamery zjednodušuje na určení převodní konstanty o rozměru mm/px (tzv. měřítka, rozlišení). Pixely kamery jsou čtvercové, proto je měřítko v obou osách kamery stejné.

Prováděním dalších typů kalibrace, např. pro odstranění zkreslení kamery (poduškovitého nebo soudkovitého), se pro dobré vlastnosti snímků z kamery není nutné v této práci zabývat.

Teoreticky lze získat měřítko výpočtem, bez provádění jakýchkoliv experimentů, je k tomu však nutná detailní znalost parametrů kamery a objektivu. Na první pohled se nabízí použití rovnic 2.1 a 2.2 pro získání velikosti zorného pole a pak prosté podělení výsledku rozlišením senzoru kamery. Tento přístup však není správný, rovnice 2.1 je pro naznačené použití platná pouze při zaostření objektivu do nekonečna. Při jiném zaostření (v praxi běžném) se ohnisková vzdálenost kamery mění a vztah se komplikuje. Častější a přesnější je určování měřítka experimentálně pořízením snímku objektu známé velikosti. S rostoucí velikostí tohoto objektu klesá chyba stanovení měřítka, proto je doporučeno měřit přímo zorné pole – např. pořízením snímku svinovacího metru umístěného v rovině snímání. Měřítko je získáno podělením neměřené velikosti zorného pole rozlišením senzoru ve směru měření. Je důležité zdůraznit, že takto získané měřítko je platné pouze pro použitou vzdálenost roviny snímání od kamery.

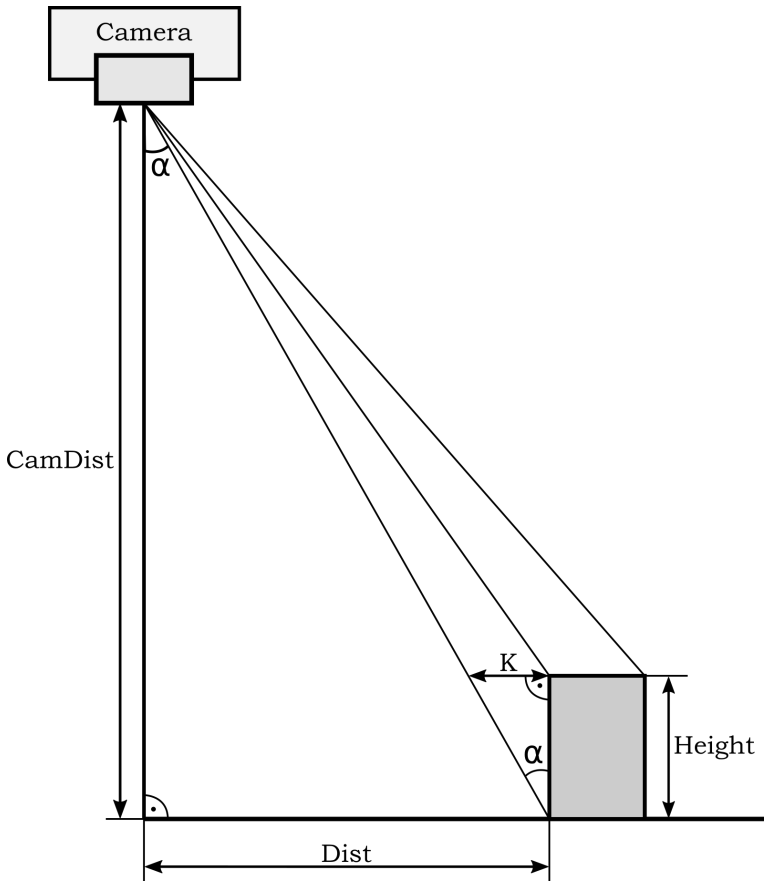
2.3.2 Korekce zkreslení daného perspektivou

Obrobky přijíždějící pod kameru na páse povětšinou nebudou v ose snímání kamery. A protože mají nenulovou výšku (typicky okolo 50 mm), projeví se na snímcích zkreslení dané perspektivou – čím více na kraji zorného pole kamery jsou, tím více se zkreslení projevuje. Zkreslením je zamýšlen fakt, že jsou na snímku viditelné boční stěny obrobku, které Blob analýza nerozezná od horní stěny a za nalezený objekt označí celý obrobek, jak je naznačeno na následujících obrázcích 2.1 a 2.2.

Obr. 2.1: Snímek obrobku na kraji FOV

Obr. 2.2: Výsledek Blob analýzy obrobku

Z uvedených obrázků je patrné, že je nutné korigovat jednak rozměry obrobku, jednak jeho pozici. Situace je znázorněná na obrázku 2.3, který bude následně použit k odvození rovnice pro zjištění velikosti korekce K.



Obr. 2.3: Znázornění výpočtu zkreslení daného perspektivou

Rovina snímání se nachází ve vzdálenosti $CamDist$ (mm) od kamery. Pro jednoduchost předpokládejme, že jsou osy obdélníkového obrobku rovnoběžné s osami čipu kamery. Vzdálenost bližší hrany obrobku od osy snímání je značena $Dist$ (mm) a na čip kamery se zobrazí pod úhlem α. Výška obrobku je značena $Height$ (mm). Pro velikost úhlu α platí:

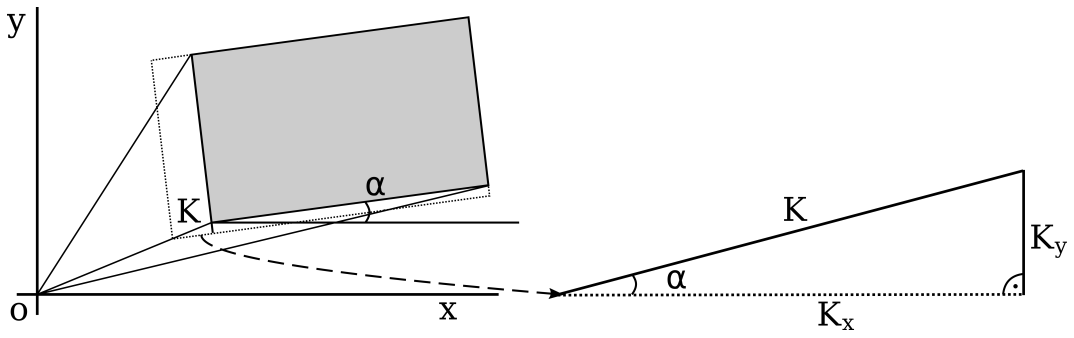
$$\alpha = \arctan\left(\frac{CamDist}{Dist}\right) \ [^\circ]. \tag{2.7}$$

Díky podobnosti trojúhelníků lze zjistit velikost korekce následovně:

$$K \ = Height \cdot \tan(\alpha) = Height \cdot \tan\left(\arctan\left(\frac{CamDist}{Dist}\right)\right) \ [mm] \tag{2.8}$$

Po spočtení velikosti korekce K je nutno odečíst ji od rozměrů obrobku zjištěných Blob analýzou a určit novou pozici obrobku. Pro tento účel je v obecném případě

nutné provést rozklad K do směrů os x a y. Velikost složek korekce K_x a K_y závisí na natočení obrobku – úhlu α na obrázku 2.4.



Obr. 2.4: Znázornění rozkladu korekce K do složek K_x a K_y

Velikost složek K_x a K_y je pro případ zachycený na obrázku 2.4 následující:

$$K_x = K \cdot \cos(\alpha)\ [mm], \qquad K_y = K \cdot \sin(\alpha)\ [mm]. \tag{2.9}$$

Omezujícím faktorem pro praktické využití odvozeného vztahu 2.8, potažmo 2.9, je nutná znalost výšky obrobku *Height*. Ta musí být měřena nebo získávána jiným způsobem – například z nadřazeného systému, který zná rozměry obrobků, jež budou na pás umístěny.

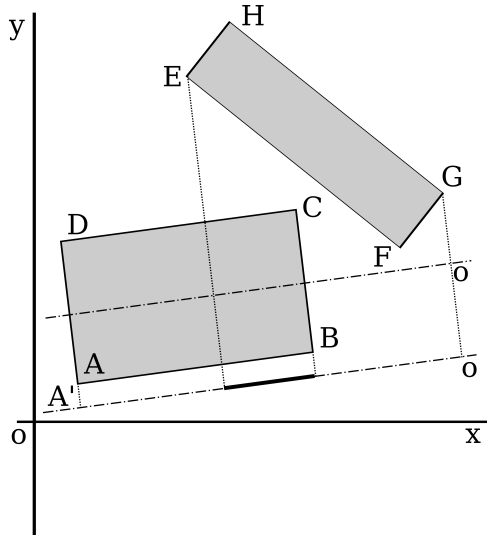
2.3.3 Detekce kolize

Detekcí kolize je zamýšlen proces, který vede k určení obrobků, jež lze odebrat robotem, aniž by došlo ke kolizi chapadla robota s jiným obrobkem. Obdélníkové obrobky budou uchopovány vždy v jejich středu kolmo na osu danou delší stranou. V této oblasti musí být dostatečný prostor, aby se do něho i s jistou rezervou vešlo otevřené chapadlo robota. Zmíněný prostor je vymezen šířkou a maximálním rozevřením chapadla a tvoří obdélník kolmý na obrobek. Problém detekce kolize při uchopování daného obrobku tedy spočívá v ověření, že tento obdélníkový prostor nezasahuje do žádného jiného obrobku.

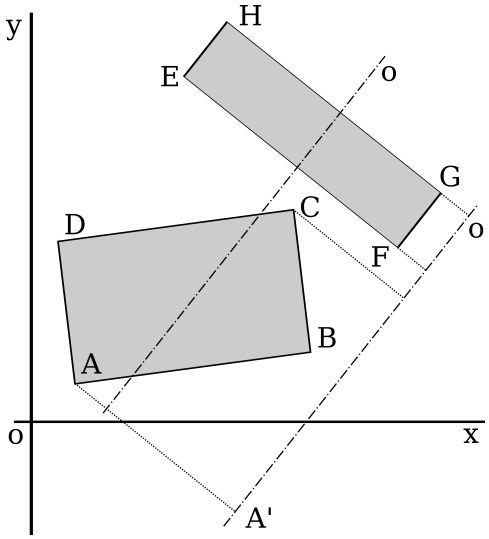
Základní otázka tedy zní, jak určit, zda mají dva obdélníky nějaký společný bod. K tomuto problému lze přistoupit vícero způsoby, elegantním řešením je využití metody SAT (*Separating Axis Theorem*, teorém o dělicí ose). SAT funguje na principu hledání osy, na které nedochází k průniku projekcí objektů, jejichž kolize je zkoumána. Pokud alespoň jedna taková osa existuje, pak objekty v kolizi nejsou – a naopak, pokud žádnou takovou osu nelze nalézt, kolidují.

Kolize obrobku a chapadla je detekována v rovině snímání – jedná se tedy o problém řešený v dvoujrozměrném prostoru. Protože se jedná o dva obdélníky, z nichž

každý má dvě osy, stačí zjistit, zda dochází k průniku projekcí na těchto čtyřech osách. Osa, na níž k průniku nedochází, je ona *separating axis* – dělicí osa, díky níž lze říci, že obdélníky v kolizi nejsou.



Obr. 2.5: Dělicí osa nenalezena



Obr. 2.6: Dělicí osa nalezena

Na obrázku 2.5 je znázorněn případ, kdy na zkoumané ose dochází k průniku projekcí obdélníků. Na obrázku 2.6 je oproti tomu situace jiná – k průniku projekcí nedochází a dělicí osa je nalezena. Lze tedy říci, že obdélníky nemají žádný společný bod a nejsou tedy v kolizi. Pro přehlednost byly osy posunuty mimo obdélníky.

Matematickou formulaci předchozích úvah provedu na příkladě z obrázku 2.5. Prvním krokem je volba osy, na níž bude provedena projekce, zde je vybrána osa obdélníku $ABCD$. Projekce vrcholu A na zvolenou osu se rovná projekci vrcholu D, to samé platí pro dvojici vrcholů B a C. Je tedy zvolen jeden z vrcholů dvojice A a D (zde A) a stanoveny jeho souřadnice ze známé pozice a rozměrů obdélníku. Poté je možno vypočítat projekci bodu A na bod A' ležící na zvolené ose O. To je možné jednoduchým výpočtem skalárního součinu bodu A a jednotkového vektoru $O/|O|$. Obdobně je získána projekce bodů B, E, a G. Pro ověření toho, zda je zvolená osa dělicí, zbývá porovnat výsledky skalárních součinů – stanovit pro každý z obdélníků interval daný projekcemi jeho vrcholů a ověřit, zda dochází k průniku těchto intervalů. Stejný postup je opakován pro zbylé osy, dokud nejsou projity všechny čtyři nebo dokud není některá z nich označena za dělicí.

Výše uvedený výpočet lze pro potřeby SAT zjednodušit. Není nutné počítat jednotkový vektor zvolené osy $O/|O|$, pro výpočet projekce lze použít přímo vektor O. Dochází tím sice ke změně měřítka, v němž vychází projekce vrcholů obdélníků, nemění to ale výsledek; zda dochází průniku intervalů. Tím odpadá nutnost opakovaně počítat velikost vektoru $|O|$, což je výpočetně poměrně složitá operace.

3 Realizace programového řešení

Na programové řešení, kterému se v této práci věnuji, je od počátku kladen požadavek na vysokou míru univerzálnosti. Je žádoucí, aby ho bylo možné bez větších úprav použít i v jiných podobných aplikacích. Rozhodl jsem se proto dát mu následující podobu. V první řadě jsou jeho součástí programové moduly. Z nich nejvýznamnější místo zaujímá modul k obsluze Smart Sensoru, který je nezávislý na daném projektu. Další moduly jsou na konkrétním projektu závislé a slouží například k předávání dat nebo obsluze uživatelské vizualizace. Nad programovými moduly je vytvořena knihovna nazvaná „Vision Template“ (v programu zkratka `VITemplate`), jež obsahuje funkce realizující všechny jednotlivé dílčí funkcionality popsané v kapitole 2. Příkladem může být například funkce pro korekci zkreslení daného perspektivou nebo funkce pro detekci kolize. Řešení je tedy do značné míry modulární.

Z důvodu efektivní kompilace a širokého rozšíření byl zvolen programovací jazyk ANSI C. Automation Studio umožňuje programovat řídicí prostředky B&R i v tomto jayzce, který není součástí normy IEC 61131-3 pro PLC. Jazyk ANSI C je též využíván ve zbylých částech projektu, jehož má být má práce součástí. V celém programu je používán výhradně anglický jazyk, jak při volbě názvů funkcí a proměnných, tak v komentářích. Při programování jsem se řídil interním dokumentem stanovujicím programovací konvence.

3.1 Knihovna VITemplate

Jak již bylo naznačeno v úvodu kapitoly 3, knihovna `VITemplate` tvoří jádro této práce. Bude jí proto v této podkapitole věnována náležitá pozornost.

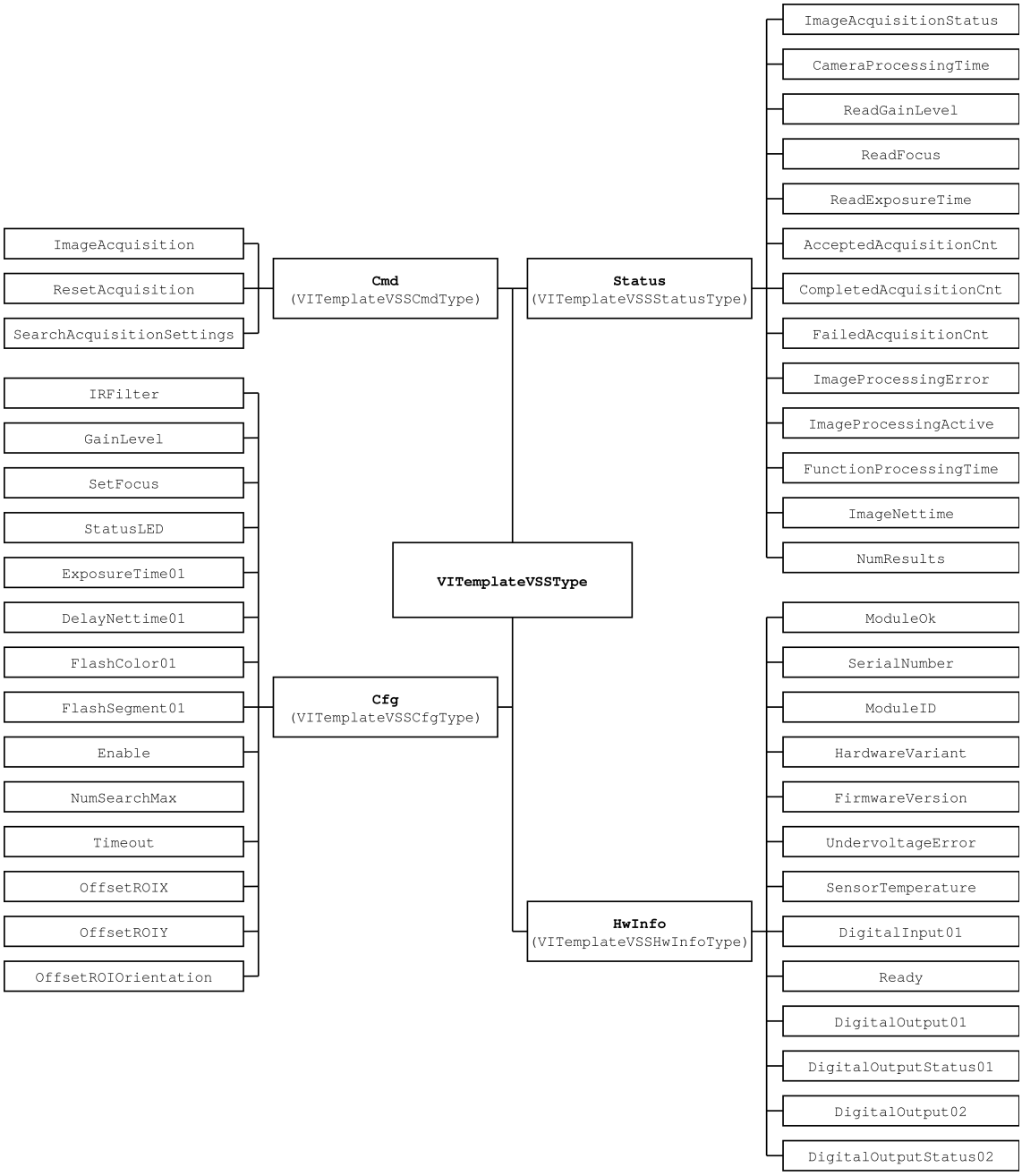
3.1.1 Struktury knihovny VITemplate

Pro snažší používání knihovny `VITemplate` bylo navrženo několik základních struktur, které jsou následně použity jako globální proměnné, jichž většina knihovních funkcí používá jako vstupní argumenty. Než přistoupím k vlastnímu popisu jednotlivých implementovaných funkcí knihovny, zaměřím se na popis těchto struktur. Budu se k nim v následujících podkapitolách často odkazovat, jejich souhrn zde tedy podpoří celkovou srozumitelnost textu.

Struktura VITemplateVSSType

První popisovanou strukturou je `VITemplateVSSType`. Oproti ostatním strukturám v této podkapitole je jedinou, která není využívána žádnou funkcí. Je však klíčovou z hlediska používání Smart Sensoru, lze pomocí ní kameru ovládat, kofigurovat nebo

zjišťovat její stav. Podkladem byly při její tvorbě mapované proměnné, které jsou skrze Powerlink cyklicky komunikovány mezi kamerou a řídicím počítačem. Lze je nalézt v I/O mapping souboru Smart Sensoru a byly zachovány jejich původní názvy. Rozhodl jsem se do struktury zahrnout tyto proměnné všechny, přestože zdaleka všechny nejsou v programu využívány, a to proto, že se jedná o knihovnu, u níž se předpokládá další, jiné nasazení a rozšíření o další, jiné funkcionality.



Obr. 3.1: Struktura `VITemplateVSSType`

Proměnné struktury byly rozděleny do čtyř skupin, jimž odpovídají podstruktury

`VITemplateVSSCmdType` s příkazy pro kameru, `VITemplateVSSCfgType`, obsahující proměnné umožňující kameru konfigurovat, `VITemplateVSSStatusType` s aktuálním statusem kamery a `VITemplateVSSHwInfoType` s informacemi o kameře z hlediska HW.

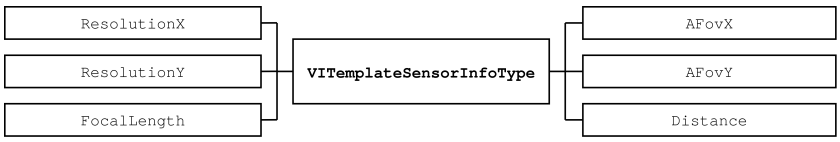
V době vzniku této práce není k dispozici oficiální distribuce Smart Sensoru, pouze jeho prototyp, proto není k dispozici ani žádná oficiální dokumentace. Z tohoto důvodu zde velmi stručně popíšu význam jednotlivých proměnných příslušného I/O mapping souboru.

`Cmd.ImageAcquisition:` příkaz pořízení snímku.
`Cmd.ResetAcquisition:` příkaz zrušení zpožděného pořízení snímku.
`Cmd.SearchAcquisitionSettings:` příkaz hledání nastavení ostření, expozice.
`Cfg.IRFilter:` aktivace/deaktivace infračerveného filtru.
`Cfg.GainLevel:` nastavení zesílení jasu pořízeného snímku.
`Cfg.SetFocus:` nastavení ostření (minimální vzdálenost až nekonečno).
`Cfg.StatusLED:` výběr barvy a módu statusových LED.
`Cfg.ExposureTime01:` nastavení expozičního času v μs.
`Cfg.DelayNettime01:` nastavení času pro zpožděné pořízení snímku v μs.
`Cfg.FlashColor01:` výběr barvy integrovaného nasvícení (RGB, Lime).
`Cfg.FlashSegment01:` aktivace/deaktivace segmentů integrovaného nasvícení.
`Cfg.Enable:` aktivace/deaktivace zvolené Vision funkce.
`Cfg.NumSearchMax:` nastavení maximálního počtu výslednků zvolené Vision funkce.
`Cfg.Timeout:` volba časového limitu pro Vision funkci v ms.
`Cfg.OffsetROIX:` nastavení posunu oblasti zájmu ve směru osy x v 1/100 px.
`Cfg.OffsetROIY:` nastavení posunu oblasti zájmu ve směru osy y v 1/100 px.
`Cfg.OffsetROIOrientation:` nastavení orientace oblasti zájmu (-179,99° až 180,00°).
`Status.ImageAcquisitionStatus:` specifikace statusu kamery.
`Status.CameraProcessingTime:` doba zpracování posledního snímku.
`Status.ReadGainLevel:` nastavená úroveň zesílení jasu pořízeného snímku.
`Status.ReadFocus:` aktuálně nastavená hodnota zaostření.
`Status.ReadExposureTime:` aktuálně nastavená doba expozice v μs.
`Status.AcceptedAcquisitionCnt:` počítadlo přijatých příkazů k pořízení snímku.
`Status.CompletedAcquisitionCnt:` počítadlo úspěšných pořízení snímku.
`Status.FailedAcquisitionCnt:` počítadlo selhaných pořízení snímku.
`Status.ImageProcessingError:` specifikace chybového statusu kamery.
`Status.ImageProcessingActive:` příznak aktuálně probíhajícího zpracování snímku.
`Status.FunctionProcessingTime:` čas trvání zvolené Vision funkce v ms.
`Status.ImageNettime:` NetTime časová známka posledního pořízeného snímku.
`Status.NumResults:` počet výsledků zvolené Vision funkce.

`HwInfo.ModuleOk`: příznak siganlizující stav modulu kamery.
`HwInfo.SerialNumber`: výrobní číslo kamery.
`HwInfo.ModuleID`: identifikační číslo HW kamery.
`HwInfo.HardwareVariant`: HW varianta kamery.
`HwInfo.FirmwareVersion`: verze FW kamery.
`HwInfo.UndervoltageError`: příznak siganlizující chybu napájení.
`HwInfo.SensorTemperature`: aktuální teplota kamery ve °C.
`HwInfo.DigitalInput01`: aktuální status digitálního vstupu (externí spoušť).
`HwInfo.Ready`: příznak signalizující připravenost kamery k použití.
`HwInfo.DigitalOutput01`: výběr funkce digitálního výstupu kamery č. 1.
`HwInfo.DigitalOutputStatus01`: status digitálního výstupu kamery č. 1.
`HwInfo.DigitalOutput02`: výběr funkce digitálního výstupu kamery č. 2.
`HwInfo.DigitalOutputStatus02`: status digitálního výstupu kamery č. 2.

Struktura VITemplateSensorInfoType

Struktura `VITemplateSensorInfoType` souvisí velmi úzce se Smart Sensorem, stejně jako ta předchozí. Oproti ní je však vázaná na konkrétní zvolenou konfiguraci kamery; rozlišení snímače, optiku. To jsou informace, které nelze vyčíst z mapovaných proměnných a je nutné zadat je manuálně. Krom rozlišení čipu `ResolutionX` a `ResolutionY` [px] a ohniskové vzdálenosti objektivu `FocalLength` [mm] jsou zde tři proměnné vázané na uživatelské měření. Proměnné `AFovX` a `AFovY` [rad] nesou informaci o úhlovém zorném poli kamery v obou směrech snímače, jehož velikost lze zjistit postupem popsaným v části 2.1. Do proměnné `Distance` [mm] je zapsána vzdálenost kamery od snímané scény.



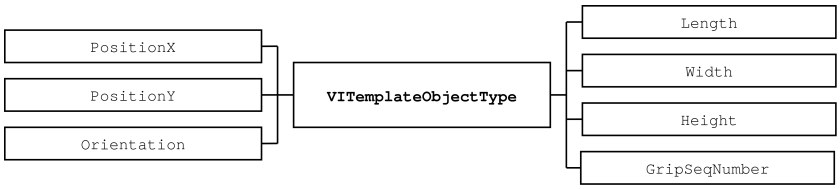
Obr. 3.2: Struktura `VITemplateSensorInfoType`

Struktura VITemplateObjectType

Každá proměnná typu struktura `VITemplateObjectType` reprezentuje v programu jeden z nalezených objektů. Dalo by se říci, že se jedná o virtuální otisk objektu nesoucí všechny informace podstatné pro to, aby bylo možné zpracovat požadavek na jeho odebrání robotickým ramenem. Jsou to informace o pozici `PositionX`, `PositionY` a rozměrech objektu `Length`, `Width`, `Height` v milimetrech. Nulová pozice je v levém horním rohu snímku, rozměr x roste směrem doprava, y směrem dolů.

Dále je zde uložena informace o jeho natočení `Orientation` v intervalu 0 až 180° a možnosti uchopení `GripSeqNumber`. Pokud je tato poslední proměnná nulová, není možné objekt robotem uchopit – buď okolo něho není dostatečný prostor pro chapadlo robota (nebezpečí kolize), nebo to ještě nebylo prověřeno. V případě nenulové hodnoty tato znamená pořadové číslo odebrání stanovené algoritmem hledajícím pořadí odebrání optimální z hlediska počtu odebratelných objektů.

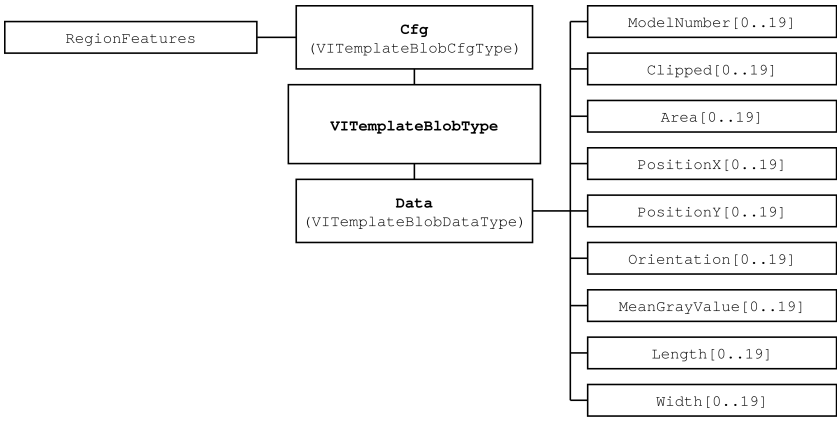
V programu je používáno pole těchto struktur o velikosti 20 prvků, což je maximální počet objektů, který může Smart Sensor při vyhodnocení jednoho snímku pomocí Blob analýzy vrátit.



Obr. 3.3: Struktura `VITemplateObjectType`

Struktura VITemplateBlobType

`VITemplateBlobType` je název struktury, která konfiguruje Blob Vision funkci, a která obsahuje její výsledky. Všechny její proměnné jsou mapovány na I/O mapping soubor Smart Sensoru. U Blob funkce lze konfigurovat jediné – zda mají být při vyhledávání objektů ve snímku brány v potaz tzv. region features. Jedná se o příznaky *Rectangularity*, *Circularity* a *Anisometry* popsané v části 1.3.1. Tento příkaz je vnořen do podstruktury `VITemplateBlobCfgType` z důvodu kompatibility s ostatními Vision funkcemi, pro které byly vytvořené obdobné, stejně členěné struktury, a které mají proměnných pro konfiguraci více.



Obr. 3.4: Struktura `VITemplateBlobType`

Výsledky Blob analýzy pak jsou ukládány do podstruktury `VITemplateBlobDataType`, jež obsahuje devět dvacetiprvkových polí, z nichž lze o každém z nalezených objektů vyčíst údaj o čísle jeho modelu `ModelNumber` [-], zda je ve snímku celý, nebo je oříznut `Clipped` [-], jaká je jeho plocha `Area` [px], pozice `PositionX` a `PositionY` [px], přičemž jako vztažný bod pro určení pozice je brán levý horní roh obrazu a x roste směrem doprava, y směrem dolů, natočení `Orientation` v intervalu -179,99 až 180°, střední jasová hodnota `MeanGreyValue` v intervalu 0 až 255, a konečně délka a šířka `Length` a `Width` [px].

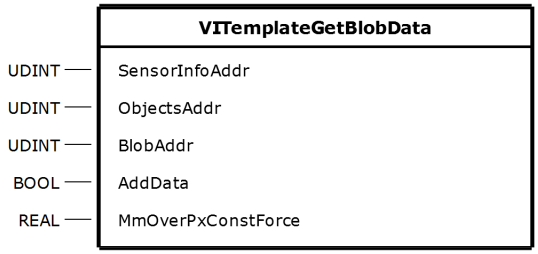
Struktura VITemplateSvgInfoType

Poslední strukturou knihovny `VITemplate` používanou v této práci je `VITemplateSvgInfoType`. Pro vizualizaci výsledků rozpoznávání je pro uživatelské HMI tvořen SVG textový řetězec. Tato struktura slouží jako vstupní parametr funkce, která SVG řetězec tvoří a obsahuje informace, které jsou k tomu zapotřebí. Základními informacemi jsou ukazatel na řetězec `StringAddr` a jeho délka `StringLength`, dalšími pak velikost vykreslovací plochy SVG `SizeX` a `SizeY`, barva SVG kresby `Color` (0: černá, 1: červená, 2: zelená, 3: modrá, 4: žlutá, 5: cyan, 6: magenta, 7: oranžová), zda má být vykreslen středový kříž nalezených objektů `DrawCross` a zda má být u nalezeného objektu napsáno jeho číslo (index v poli struktur popsaných výše v části 3.1.1) `WriteObjNumber`.

Obr. 3.5: Struktura `VITemplateSvgInfoType`

3.1.2 Funkce VITemplateGetBlobData

První funkcí, která je volána bezprostředně po pořízení a vyhodnocení snímku, je `VITemplateGetBlobData`, která vyčte informace o nalezených objektech z mapovaných proměnných Smart Sensoru a uloží je do pole struktur `VITemplateObjectType`, se kterým je pracováno dále v programu.



Obr. 3.6: Funkce `VITemplateGetBlobData`

Argument	**D. typ**	**Význam argumentu**
`SensorInfoAddr`	`UDINT`	Adresa proměnné typu `VITemplateSensorInfoType`
`ObjectsAddr`	`UDINT`	Adresa proměnné typu `VITemplateObjectType`
`BlobAddr`	`UDINT`	Adresa proměnné typu `VITemplateBlobType`
`AddData`	`BOOL`	Příznak značící přepis (log. 0) nebo přidání dat (log. 1)
`MmOverPxForce`	`REAL`	Vnucená převodní konstanta mm/px (pokud je > 0)
Návr. hodnota	`USINT`	Počet vyčtených objektů (< 200), kód chyby (≥ 200)

Tab. 3.1: Argumenty a návratová hodnota funkce `VITemplateGetBlobData`

Pozice a rozměry nalezených objektů jsou vyčteny v setinách pixelů, uvnitř funkce jsou převedeny na hodnoty v milimetrech (kalibrace kamery). To je provedeno jedním ze způsobů popsaných v části 2.3.1. Pokud je nenulový argument funkce `MmOverPxConstForce` (převodní konstanta o rozměru mm/px), je hodnota v milimetrech získána jednoduchým násobením. Pokud tato možnost využita není, je převodní konstanta získána výpočtem rovnice 2.2. Cílem je získat velikost průmětu objektu do roviny rovnoběžné s rovinou snímání, která je nad rovinou snímání umístěna ve výšce dané výškou objektu. Tato skutečnost musí být zohledněna v případě použití experimentálně stanovené převodní konstanty. Jak bylo uvedeno v části o kalibraci kamery již zmiňované výše, takto zjištěná konstanta je platná pouze pro danou vzdálenost od kamery.

V mapované proměnné, která obsahuje šířku nalezených objektů, je uložen vždy jeho delší rozměr, což je poněkud matoucí, proto je delší rozměr v rámci struktury `VITemplateObjectType` uložen do proměnné pro délku. Rozsah možné orientace nalezeného objektu -179,99° až 180° je omezen pouze na kladnou část intervalu, protože je pro obdélníkové objekty postačující.

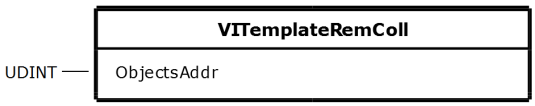
Argument funkce s názvem `AddData` umožňuje volbu mezi tím, zda vyčítaná data získaná Blob analýzou nahradí starší data v poli struktur typu `VITemplateObject-`

`Type`, nebo budou k starším datům přidána (log. 1). Přidávání dat může být využito při vícenásobném snímání scény s různým nastavením expozice (např. barvou nasvícení) pro sloučení výsledků několika vyhodnocení těchto snímků.

Návratovou hodnotou funkce je kladné číslo mající význam počtu vyčtených a přidaných objektů (< 200, prakticky však ≤ 20 kvůli maximálnímu počtu výsledků Blob Vision funkce) nebo význam chybového kódu (≥ 200). Funkce `VITemplateGetBlobData` může vracet následující chybové kódy: `viTEMPLATE_ERR_MISSING_INPUT` (200) nebo `viTEMPLATE_ERR_WRONG_INPUT` (201). První z jmenovaných chyb značí, že některý z povinných argumentů je nulový, druhá pak, že některý z argumentů nabývá chybé hodnoty.

3.1.3 Funkce VITemplateRemColl

Po naplnění pole struktur `VITemplateObjectType` aktuálními výsledky Blob analýzy předchozí funkcí je možné použít funkci `VITemplateRemColl` (kde *„RemColl“* je zkratkou pro *remove colliding*), která z tohoto pole odebere objekty, které jsou v kolizi. Vzhledem ke způsobu, jakým Blob Vision funkce vyhledává objekty ve snímku, se kolidující výsledky vyskytnou tehdy, vyhovuje-li jeden objekt dvěma či více modelům (duplicita). Při testování bylo zjištěno, že má použití více modelů pro stejný objekt praktický význam. Na příkladě obdélníkových obrobků bylo toto potřebné například z důvodu nerovnoměrného nasvícení scény (stejný objekt je na snímku různě tmavý), povrchové variability (různá tmavost materiálu, rez) nebo zkreslení daného perspektivou (viditelné boční stěny způsobují změnu tvaru). Vytvořit jediný model, jenž by obsáhnul všechny vlivy působící na podobu objektu na snímku, je velmi náročné. Daleko jednodušším způsobem je vytvořit postupně modelů více; přidávat další vždy, když nedojde k nalezení objektu, dokud není dosaženo požadované kvality vyhodnocení.



Obr. 3.7: Funkce `VITemplateRemColl`

Argument	**Dat. typ**	**Význam argumentu**
`ObjectsAddr`	`UDINT`	Adresa proměnné typu `VITemplateObjectType`
Návr. hodnota	`USINT`	Počet odebraných objektů (< 200), chyba (≥ 200)

Tab. 3.2: Argumenty a návratová hodnota funkce `VITemplateRemColl`

Pro stanovení toho, zda dva objekty kolidují, je využíváno SAT (teorému o dělicí ose) popsaném v části 2.3.3. Jeho implementace má podobu statické funkce `VITemplateCollision`.

Funkce má návratovou hodnotu s významem počtu odebraných objektů (< 200, prakticky však ≤ 20 kvůli délce pole struktur `VITemplateObjectType`) nebo význam chybového kódu (≥ 200). Mezi možné chybové kódy patří: `viTEMPLATE_ERR_MISSING_INPUT` (200) nebo `viTEMPLATE_ERR_WRONG_INPUT` (201).

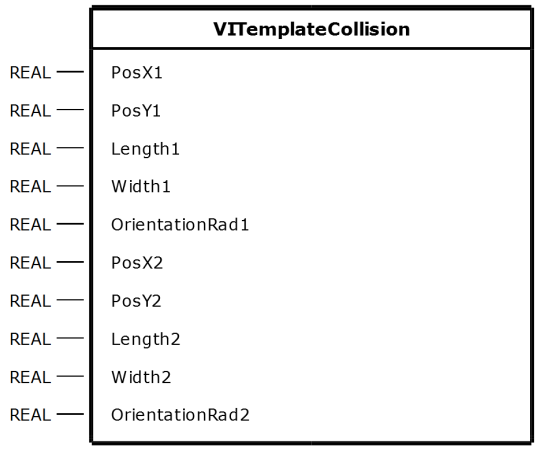
Statická funkce VITemplateCollision

Implementací SAT, teorému o dělicí ose, popsaném v části 2.3.3 je statická funkce `VITemplateCollision`. Oproti klasické metodě výpočtu SAT je využito i zjednodušení popsaném na stejném místě, které vede ke snížení výpočetní náročnosti kódu.

V programovacím jazyce C jsou funkce defaultně globální. Při programování PLC od B&R to znamená, že je lze využívat ve všech částech programu (v jiných funkcích, funkčních blocích a programových modulech). Přístup ke statickým funkcím je oproti tomu omezen pouze na soubor, v němž jsou deklarovány. Této vlastnosti je zde využíváno za účelem skrytí funkce `VITemplateCollision` před uživately knihovny VITemplate z důvodu větší přehlednosti. Není předpokládáno její používání mimo funkce knihovny VITemplate a její přítomnost mezi nimi by byla zbytečná a matoucí.

Pozice a velikost obou objektů může být funkci předáná v libovolných jednotkách, tyto jednotky však musí být napříč všemi argumenty shodné. Vyjímku samozřejmě tvoří orientace objektů, která musí být předána v radiánech.

Funkce `VITemplateCollision` vrací hodnotu 0 v případě, že předané obdélníkové objekty nekolidují a hodnotu 1 v případě, že ano. Návratová hodnota ≥ 200 značí chybu, a to buď `viTEMPLATE_ERR_MISSING_INPUT` (200) nebo `viTEMPLATE_ERR_WRONG_INPUT` (201).



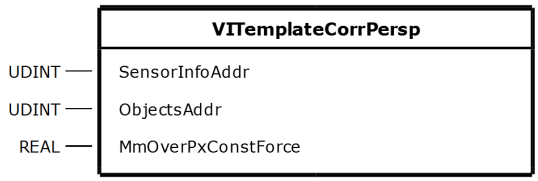
Obr. 3.8: Statická funkce `VITemplateCollision`

Argument	Dat. typ	Význam argumentu
PosX1	REAL	Pozice prvního objektu ve směru osy x
PosY1	REAL	Pozice prvního objektu ve směru osy y
Length1	REAL	Delší rozměr prvního objektu
Width1	REAL	Kratší rozměr prvního objektu
OrientationRad1	REAL	Orientace prvního objektu v radiánech
PosX2	REAL	Pozice druhého objektu ve směru osy x
PosY2	REAL	Pozice druhého objektu ve směru osy y
Length2	REAL	Delší rozměr druhého objektu
Width2	REAL	Kratší rozměr druhého objektu
OrientationRad2	REAL	Orientace druhého objektu v radiánech
Návr. hodnota	USINT	Objekty nekolidují (0), kolidují (1), chyba (geq 200)

Tab. 3.3: Argumenty a návratová hodnota statické funkce VITemplateCollision

3.1.4 Funkce VITemplateCorrPersp

V době, kdy je pole struktur VITemplateObjectType naplněno nekolidujícími výsledky Blob analýzy, je možno doplnit údaj o výšce obrobků. Pokud nemají všechny nalezené objekty stejnou výšku, je nutné zajisit, aby byla tato informace správně dodána nadřazeným systémem. Tím se tato práce nezabývá, výška obrobků je pro danou várku vždy stejná a předem známá. Po stanovení výšky je možné provést korekci geometrického zkreslení podle části 2.3.2 voláním funkce VITemplateCorrPersp.



Obr. 3.9: Funkce VITemplateCorrPersp

Argument	Dat. typ	Význam argumentu
SensorInfoAddr	UDINT	Adresa proměnné typu VITemplateSensorInfoType
ObjectsAddr	UDINT	Adresa proměnné typu VITemplateObjectType
MmOverPxForce	REAL	Argument pro vnucení převodní konstanty mm/px
Návr. hodnota	USINT	Korekce provedena (0), kód chyby ($\geq$ 200)

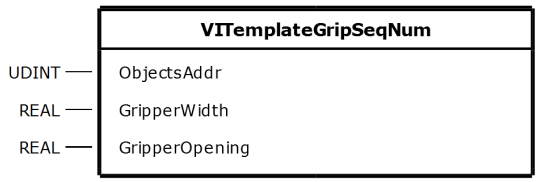
Tab. 3.4: Argumenty a návratová hodnota funkce VITemplateCorrPersp

Aby bylo možné použít vztahy 2.8 a 2.9 odvozené v části 2.3.2 odkazované již výše, je nutné vzít v potaz několik skutečností. V první řadě je to fakt, že se zkreslení projevuje tehdy, když je obrobek umístěn mimo osu snímání, která se nachází ve středu snímku. Z tohoto hlediska je souřadný systém s počátkem v levém horním rohu snímku a směrem osy y opačným, než je běžná konvence, nevhodný. Souřadný systém je tedy transformován – jeho počátek posunut do středu snímku a směr osy y otočen. Dále jsou vypočteny souřadnice jednotlivých vrcholů obdélníku daného pozicí, rozměry a orientací nalezeného objektu. Jako bod, pro který je počítána korekce K, je zvolen vrchol, který je nejblíže novému počátku souřadného systému – ose snímání. Pokud je tento vrchol v jiném, než prvním kvadrantu, je objekt horizontálně, případně vertikálně nebo oběma způsoby, zrcadlen do vedlejšího kvadrantu tak, aby v prvním kvadrantu nejbližší vrchol byl. Tak je problém zjednodušen na jediný kvadrant. Zde jsou podle toho, o jaký vrchol se jedná, upraveny rozměry objektu a vypočtena jeho nová pozice. Poté je objekt zrcadlen zpět do kvadrantu, ve kterém se původně nacházel, pokud je to nutné. Na závěr je provedena inverzní transformace souřadného systému.

Návratovou hodnotou funkce `VITemplateCorrPersp` je 0 v případě, že byla korekce úspěšně provedena nebo číslo ≥ 200 značící chybu. Konkrétně to mohou být opět chyby `viTEMPLATE_ERR_MISSING_INPUT` (200) nebo `viTEMPLATE_ERR_WRONG_INPUT` (201).

3.1.5 Funkce VITemplateGripSeq

Po volání funkcí popsaných v předcházejícím textu je možné přistoupit k finálnímu kroku vyhodnocení, kterým je stanovení pořadí odebrání nalezených objektů robotem. Současně s tímto je nutné ověřovat, že je v okolí odebíraného objektu dostatečný prostor pro chapadlo robota. Funkce `VITemplateGripSeq` řeší obojí. Krom pole struktur `VITemplateObjectType`, ve kterém jsou pozice a rozměry všech nalezených objektů, je pro stanovení pořadí nutné znát šířku chapadla robota (argument `GripperWidth`) a jeho velikost v druhém směru v plném rozevření (argument `GripperOpening`). Oba tyto rozměry jsou v milimetrech a je doporučeno k nim přidat dostatečnou rezervu. Podle nich je totiž rozhodováno o tom, zda je okolo objektu dostatečný prostor a je bezpečné pokusit se o jeho odebrání.



Obr. 3.10: Funkce `VITemplateGripSeq`

Argument	Dat. typ	Význam argumentu
`ObjectsAddr`	`UDINT`	Adresa proměnné typu `VITemplateObjectType`
`GripperWidth`	`REAL`	Šířka chapadla robota v mm
`GripperOpening`	`REAL`	Rozpětí otevřeného chapadla robota v mm
Návr. hodnota	`USINT`	Počet neuchopitelných obj. (< 200), chyba (≥ 200)

Tab. 3.5: Argumenty a návratová hodnota funkce `VITemplateGripSeq`

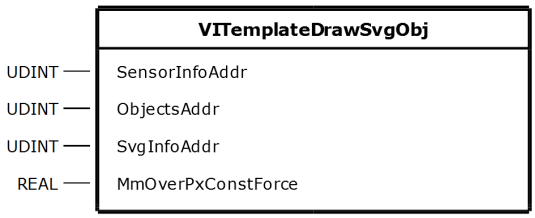
Funkce `VITemplateGripSeq` pracuje ve dvou krocích, které jsou podle potřeby několikrát opakovány. V prvním kroku jsou postupně projity všechny nalezené objekty, ze kterých již byly funkcí `VITemplateRemColl` vyfiltrovány všechny kolidující mezi sebou, a s každým z nich je provedena následující procedura. Z pozice a orientace objektu je určeno, kde by se nacházelo chapadlo robota při úchopu. Obdélníkový objektu je uchopován ve svém těžišti, proto je pozice shodná. Orientace chapadla je pak kolmá na orientaci obrobku. K těmto údajům jsou přidány argumenty funkce `GripperWidth` a `GripperOpening`, čímž vzniká obdélník, jehož kolizi s ostatními objekty ve scéně lze vyhodnotit statickou funkcí `VITemplateCollision` popsanou již dříve v části 3.1.3. Pokud je s nějakým objektem v kolizi, je jeho index, spolu s indexem právě uchopovaného objektu, zapamatován (uložen do *bufferu*). Po aplikaci popsané procedury na všechny nalezené objekty je získán seznam (*buffer*) kolizí. Maximální počet dvojic indexů v tomto seznamu je dán konstantou `viCOLLISION_BUFFER_MAX_INDEX`. V druhém kroku jsou přiřazena pořadová čísla odebrání `GripSeqNumber` (začínající od jedničky) všem objektům, které nejsou na seznamu kolizí. Poté může být opakován první krok s tím, že je předpokládáno, že už jsou všechny objekty, které bylo možno odebrat (mají nenulové `GripSeqNumber`), odebrané. Je vytvořen nový seznam kolizí, který může být kratší, než v přechozí iteraci, protože mohl být nějaký objekt v kolizi odebrán. V tom případě mohou být v druhém kroku některým z dalších objektů přiřazena pořadová čísla. Tento postup je iterativně opakován, dokud je to smysluplné, tedy dokud dochází ke snižování počtu objektů kolidujících s chapadlem robota.

Je důležité zmínit, že výše popsané řešení detekce kolize chapadla robota s objekty předpokládá, že krom nalezených objektů ve scéně snímání nic jiného není.

Návratovou hodnotou funkce `VITemplateGripSeq` je počet objektů, které nelze uchopit, nebo v případě, že se jedná o číslo ≥ 200, je to chybový kód. Chyby mohou být následující `viTEMPLATE_ERR_MISSING_INPUT` (200), `viTEMPLATE_ERR_WRONG_INPUT` (201) nebo `viTEMPLATE_ERR_TOO_MANY_COLL` (203). Význam poslední jmenované chyby je, že bylo nalezeno příliš mnoho kolizí. Připravený buffer o velikosti dané konstantou `viCOLLISION_BUFFER_MAX_INDEX` nebyl dostatečný.

3.1.6 Funkce VITemplateDrawSvgObj

Zbývající funkce jsou určeny k použití v uživatelské vizualizaci. První z nich je `VITemplateDrawSvgObj`, funkce, která vytvoří textový řetězec ve formátu značkovacího jazyka SVG (podle otevřeného standartu W3C [19]) použitelného pro vektorovou grafiku na webových stránkách. Na výrobcích B&R je v současnosti upřednostňována tvorba vizualizace skrze mapp komponentu *mapp View*, která je posléze zobrazována ve webovém prohlížeči. V mapp View je implementován *Paper widget*, nástroj pro zobrazení SVG elementů. Proto je tento formát vhodný pro použití k vykreslení výsledků provedeného vyhodnocení.



Obr. 3.11: Funkce `VITemplateDrawSvgObj`

Argument	**Dat. typ**	**Význam argumentu**
`SensorInfoAddr`	`UDINT`	Adr. proměnné typu `VITemplateSensorInfoType`
`ObjectsAddr`	`UDINT`	Adresa proměnné typu `VITemplateObjectType`
`SvgInfoAddr`	`UDINT`	Adresa proměnné typu `VITemplateSvgInfoType`
`MmOverPxForce`	`REAL`	Argument pro vnucení převodní konstanty mm/px
Návr. hodnota	`USINT`	SVG textový řetězec vytvořen (0), kód chyby ($\geq$ 200)

Tab. 3.6: Argumenty a návratová hodnota funkce `VITemplateDrawSvgObj`

Argumenty funkce jsou využity následujícím způsobem. Struktura typu `VITemplateSensorInfoType` je využita k výpočtu převodní konstanty mm/px, pokud však samozřejmě není využito argumentu `MmOverPxForce`. Převodní konstanta je tentokrát použita ke konverzi opačným směrem, tedy k převodu údajů o pozici a velikosti objektů v poli struktur typu `VITemplateObjectType` z milimetrů na pixely. Převedené hodnoty jsou ještě uzpůsobeny velikosti vykreslovací plochy SVG, která je funkci předána ve struktuře typu `VITemplateSvgInfoType`. V základu jsou vykresleny čtyři čáry tvořící obdélníkový obrys objektu. K tomu je využita statická funkce `VITemplateDrawSvgAppendLine` popsaná níže. Barva kresby je stanovena proměnnou `Color` struktury typu `VITemplateSvgInfoType`, na výběr jsou některé základní barvy (0: černá, 1: červená, 2: zelená, 3: modrá, 4: žlutá, 5: cyan, 6: magenta, 7: oranžová). Nastavením zbylých příznaků v této struktuře je ke kresbě možno přidat kříž

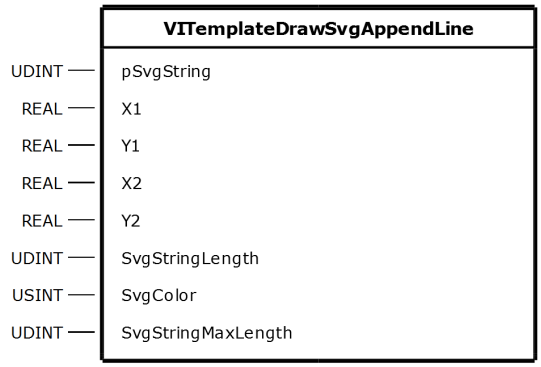
spojující úhlopříčně protilehlé vrcholy obdélníku (`DrawCross`) a číslo objektu, respektive jeho index v poli struktur typu `VITemplateObjectType` (`WriteObjNumber`). K posledně zmiňovanému je využíváno statické funkce `VITemplateDrawSvgAppendNumber` taktéž popsané v následujícím textu. Funkce před každým zápisem do řetězce na adrese dané členem `StringAddr` struktury typu `VITemplateSvgInfoType` kontroluje, zda je zde ještě pro zápis dostatečný prostor (`StringLength`), aby nedošlo k chybnému přistoupení k paměti.

Funkce vrací nulu v případě úspěšného vytvoření textového řetězce ve formátu SVG nebo číslo ≥ 200 v případě chyby. V tom případě je to `viTEMPLATE_ERR_MISSING_INPUT` (200), `viTEMPLATE_ERR_WRONG_INPUT` (201) nebo `viTEMPLATE_ERR_IN_STRING_SHORT` (202). S poslední chybou se v textu setkáváme poprvé a nese význam nedostatečné délky vstupního textového řetězce.

Statická funkce VITemplateDrawSvgAppendLine

Funkce `VITemplateDrawSvgObj` využívá ke své práci dvě statické funkce. První z nich je `VITemplateDrawSvgAppendLine`, která k SVG řetězci předanému pomocí argumentu `pSvgString` přidá SVG řetězec s čarou. Čára začíná v souřadnicích `X1` a `Y1` a končí v souřadnicích `X2` a `Y2` a má barvu danou argumentem `SvgColor`. Jak je barva volena viz popis funkce `VITemplateDrawSvgObj` 3.1.6. Argument `SvgStringLength` nese informaci o aktuální délce SVG řetězce a `SvgStringMaxLength` o jeho maximální přípustné délce. Před každým zápisem do řetězce je kontrolováno, že tato délka nebude překročena.

Návratová hodnota funkce nabývá naprosto stejného významu jako návratová hodnota předchozí funkce `VITemplateDrawSvgObj` 3.1.6.



Obr. 3.12: Statická funkce `VITemplateDrawSvgAppendLine`

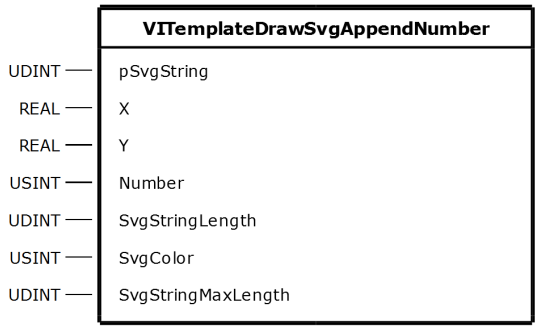
Argument	Dat. typ	Význam argumentu
pSvgString	UDINT	Adresa textového SVG řetězce
X1	REAL	Souřadnice počátečního bodu čáry ve směru osy x
Y1	REAL	Souřadnice počátečního bodu čáry ve směru osy y
X2	REAL	Souřadnice koncového bodu čáry ve směru osy x
Y2	REAL	Souřadnice koncového bodu čáry ve směru osy y
SvgStringLength	UDINT	Aktuální délka textového SVG řetězce
SvgColor	USINT	Barva SVG prvku
SvgStringMaxLength	UDINT	Maximální délka textového SVG řetězce
Návr. hodnota	UDINT	SVG textový řetězec vytvořen (0), chyba ($\geq$ 200)

Tab. 3.7: Argumenty a návr. hodnota stat. funkce `VITemplateDrawSvgAppendLine`

Statická funkce VITemplateDrawSvgAppendNumber

Obdobnou funkcionalitu jako předchozí popsaná má statická funkce `VITemplate-DrawSvgAppendNumber`, která přidává SVG řetězec s číslem k SVG řetězci předanému pomocí argumentu `pSvgString`. Je toho využíváno k vykreslení čísla nalezeného objektu, což umožní snadnou orientaci ve výsledcích analýzy snímku. Číslo nalezeného objektu je totiž zároveň indexem do tabulky, ve které jsou výsledky vypsány.

Návratová hodnota funkce má naprosto stejný význam jako návratová hodnota funkce `VITemplateDrawSvgObj` 3.1.6.



Obr. 3.13: Statická funkce `VITemplateDrawSvgAppendNumber`

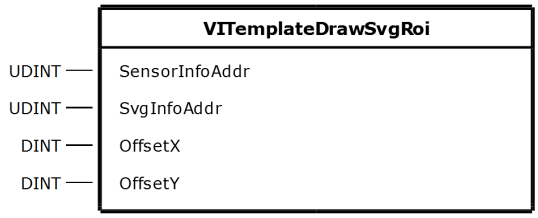
Argument	Dat. typ	Význam argumentu
pSvgString	UDINT	Adresa textového SVG řetězce
X	REAL	Souřadnice umístění čísla ve směru osy x
Y	REAL	Souřadnice umístění čísla ve směru osy y
Number	USINT	Číslo, které má být přidáno
SvgStringLength	UDINT	Aktuální délka textového SVG řetězce
SvgColor	USINT	Barva SVG prvku
SvgStringMaxLength	UDINT	Maximální délka textového SVG řetězce
Návr. hodnota	UDINT	SVG textový řetězec vytvořen (0), chyba ($\geq$ 200)

Tab. 3.8: Argumenty a návr. hodnota stat. funkce VITemplateDrawSvgAppendNumber

3.1.7 Funkce VITemplateDrawSvgRoi

Poslední funkcí, kterou zbývá popsat, je funkce VITemplateDrawSvgRoi umožňující uživateli vykreslit ve vizualizaci ROI, oblast zájmu. Jde o část snímku, která je vyhodnocována Blob analýzou, její velikost lze omezit a snížit tak celkový čas vyhodnocení. Při vykreslování ROI je využíváno statické funkce VITemplateDrawSvgAppendLine popsané již dříve v části 3.1.6.

Návratová hodnota funkce nabývá opět významu rovného návratové hodnotě funkce VITemplateDrawSvgObj 3.1.6.



Obr. 3.14: Funkce VITemplateDrawSvgRoi

Argument	Dat. typ	Význam argumentu
SensorInfoAddr	UDINT	Adresa proměnné typu VITemplateSensorInfoType
SvgInfoAddr	UDINT	Adresa proměnné typu VITemplateSvgInfoType
OffsetX	DINT	Posun ROI ve směru osy y
OffsetY	DINT	Posun ROI ve směru osy x
Návr. hodnota	USINT	SVG textový řetězec vytvořen (0), chyba ($\geq$ 200)

Tab. 3.9: Argumenty a návratová hodnota funkce VITemplateDrawSvgObj

3.2 Programový modul ViMain

Knihovna `VITemplate` představuje sice jádro této práce, nemůže ale fungovat sama o sobě. Proto zde hraje důležitou roli i programový modul `ViMain`, který mimo jiné využívá knihovních funkcí tak, aby bylo dosaženo kýžených výsledků.

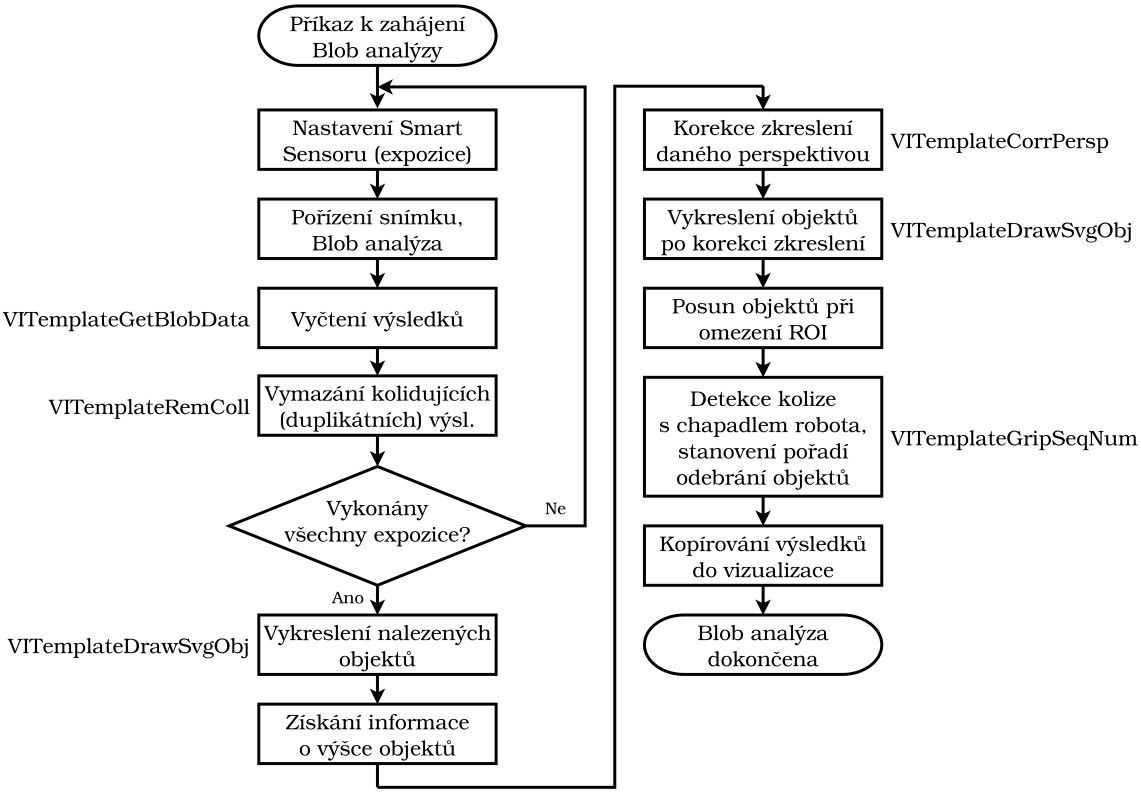
Hlavní smyčka tohoto modulu je velmi jednoduchá. Její činnost spočívá v čekání na příkaz a v jeho vykonání, pokud přijde. Příkazy, které programový modul obsluhuje, jsou následující: příkaz k zahájení Blob analýzy, příkaz k uložení konfigurace Smart Sensoru získané pomocí automatického hledání nastavení času expozice a zaostření objektivu a příkaz k vykreslení ROI, oblasti zájmu do uživatelské vizualizace.

Vykonat příkaz k zahájení Blob analýzy je oproti zbylým dvěma příkazům nepoměrně složitější, proto je tomu věnována větší část této podkapitoly. Zbylé dva příkazy budou vysvětleny v krátkosti. Uživatel se může po přípravě scény a umístění Smart Sensoru rozhodnout využít automatického hledání nastavení času expozice a zaostření objektivu. Smart Sensor pomocí jednoduchého algoritmu postaveném na mnohonásobném pořízení snímku s různým nastavením expozice nalezne takové, které vede k pořízení kvalitního snímku. Daný příkaz uloží nalezenou konfiguraci kamery a umožní jej dále používat. Zbylý příkaz, příkaz k vykreslení oblasti zájmu do uživatelské vizualizace, je vykonán v okamžiku, kdy je uživatelem změněna její velikost. Velikost je možné měnit pouze ve smyslu posunu jejího začátku z levého horního rohu o libovolný počet pixelů (omezený shora rozlišením snímku). Vlastní vykreslení ve vizualizaci je provedeno pomocí SVG řetězce vytvořeného za využití funkce `VITemplateDrawSvgRoi`.

Nyní k hlavní náplni činnosti programového modulu `ViMain`. Pro přehlednost je celý proces zanesen do vývojového diagramu, který je možné si prohlédnout na obrázku 3.15. V okamžiku, kdy je dán příkaz k zahájení Blob analýzy, jsou vymazány předchozí výsledky v poli `ViObject` (pole struktur typu `VITemplateObjectType`) a jsou zkontrolovány proměnné `ExposureTimeRed`, `ExposureTimeGreen`, `ExposureTimeBlue` a `ExposureTimeLime`. V nich mohou být uloženy expoziční časy pro snímání scény při nasvícení světlem příslušné barvy. Aby mohlo být pokračováno dále, musí alespoň jedna z proměnných nabývat nenulové hodnoty. První v pořadí, v jakém jsou v tomto textu uvedeny, obsahující nenulovou hodnotu je využita k nastavení barvy nasvícení a času expozice. Správné nastavení zaostření objektivu musí být provedeno již dříve.

Po nastavení kamery je možno pořídit snímek. Okamžitě po úspěšném pořízení snímku proběhne za využití FPGA čipu ve Smart Sensoru také nastavená Vision funkce, v tomto případě Blob analýza. Po jejím skončení jsou pomocí sběrnice Powerlink do mapovaných proměnných v PLC uloženy její výsledky. To je čas, kdy je možno volat funkci `VITemplateGetBlobData`, která výsledky z mapovaných pro-

měnných vyčte a uloží je do pole `ViObject`. Z něho jsou vzápětí voláním funkce `VITemplateRemColl` odstraněny kolidující objekty. Jak již bylo řečeno v části 3.1.3, kde je tato funkce popsána, v této fázi mohou v poli `ViObject` kolidovat pouze výsledky, kdy nějaký z nalezených objektů vyhovuje více modelům. Znovu jsou zkontrolovány proměnné s časy expozic pro různé barvy nasvícení, a pokud nenulovou hodnotu obsahuje ještě některá z nich, je podle ní nastaven Smart Sensor a pořízen další snímek. Ten je vyhodnocený stejným způsobem, jak bylo popsáno a výsledky jsou připojeny k výsledkům již existujícím v poli `ViObject`. Tento postup je opakován, dokud nejsou zkontrolovány všechny proměnné s časy expozic – je tedy možno pořídit a vyhodnotit až čtvero snímků s různou barvou nasvícení.



Obr. 3.15: Stavový diagram Blob analýzy v programovém modulu `ViMain`

Po ukončení této cyklické činnosti je zavolána funkce `VITemplateDrawSvgObj`, která kontury nalezených objektů vykreslí do uživatelské vizualizace. Následně je získána informace o výšce nalezených objektů z nadřazeného systému. V této aplikaci se jedná o jednoduchou záležitost, neboť měly všechny objekty v jedné dávce výšku shodnou. V tuto chvíli jsou již známy všechny informace potřebné k provedení korekce zkreslení daného perspektivou, může být tedy zavolána funkce `VITemplate-CorrPersp`. Po provedení korekce je podruhé volána funkce `VITemplateDrawSvgObj`, tentokrát jsou do vizualizace vykresleny objekty s opravenými rozměry a pozicemi. Ve vizualizaci je díky tomu možno pozorovat změnu, jež byla provedením korekce

zkreslení způsobeného perspektivou snímání provedena. Pokud je uživatelem proveden posun ROI, oblasti zájmu, znamená to též posun začátku souřadného systému, proto musí být případně patřičně upraveny pozice objektů.

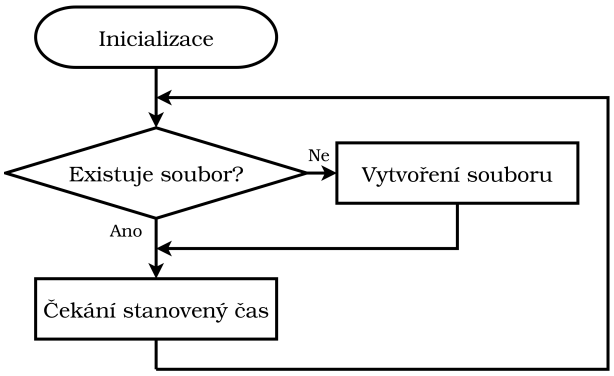
Následně je provedena detekce kolize objektů s chapadlem robota a stanovení pořadí jejich odebrání (volání funkce `VITemplateGripSeqNum`). Pořadí je stanoveno za účelem maximalizace počtu odebratelných objektů a musí být dodrženo, aby byla snížena pravděpodobnost kolize.

Tím jsou dokončeny všechny operace s výsledky Blob analýzy a data, která se nachází v poli `ViObject` mohou být použita k řízení robota. Závěrečnou prováděnou činností je jejich kopírování do polí, které mohou být zobrazeny v tabulce v uživatelské vizualizaci.

3.3 Programový modul ViImLoader

ViImLoader je programový modul, který zajišťuje, aby se v uživatelském oddílu paměti (*user partition*) PLC nacházel soubor „ImLoader.html“, který umožňuje zobrazit v uživatelské vizualizaci aktuální snímek z kamery v libovolné velikosti.

V inicializační části programového modulu je připraven textový řetězec s obsahem html souboru. V cyklické části pak je zkontrolováno, zda se v uživatelském oddílu paměti soubor nachází, či ne. Pokud ne, je soubor vytvořen. Pokud existuje, je stanovený čas počkáno (výchozí hodnotou je 60 sekund) a poté kontrola provedena znovu. Tento proces je cyklicky opakován po celou dobu běhu aplikace, je tak zajištěno, že bude soubor v případě vymazání obnoven.



Obr. 3.16: Stavový diagram hlavní smyčky programového modulu `ViImLoader`

4 Realizace vizualizace

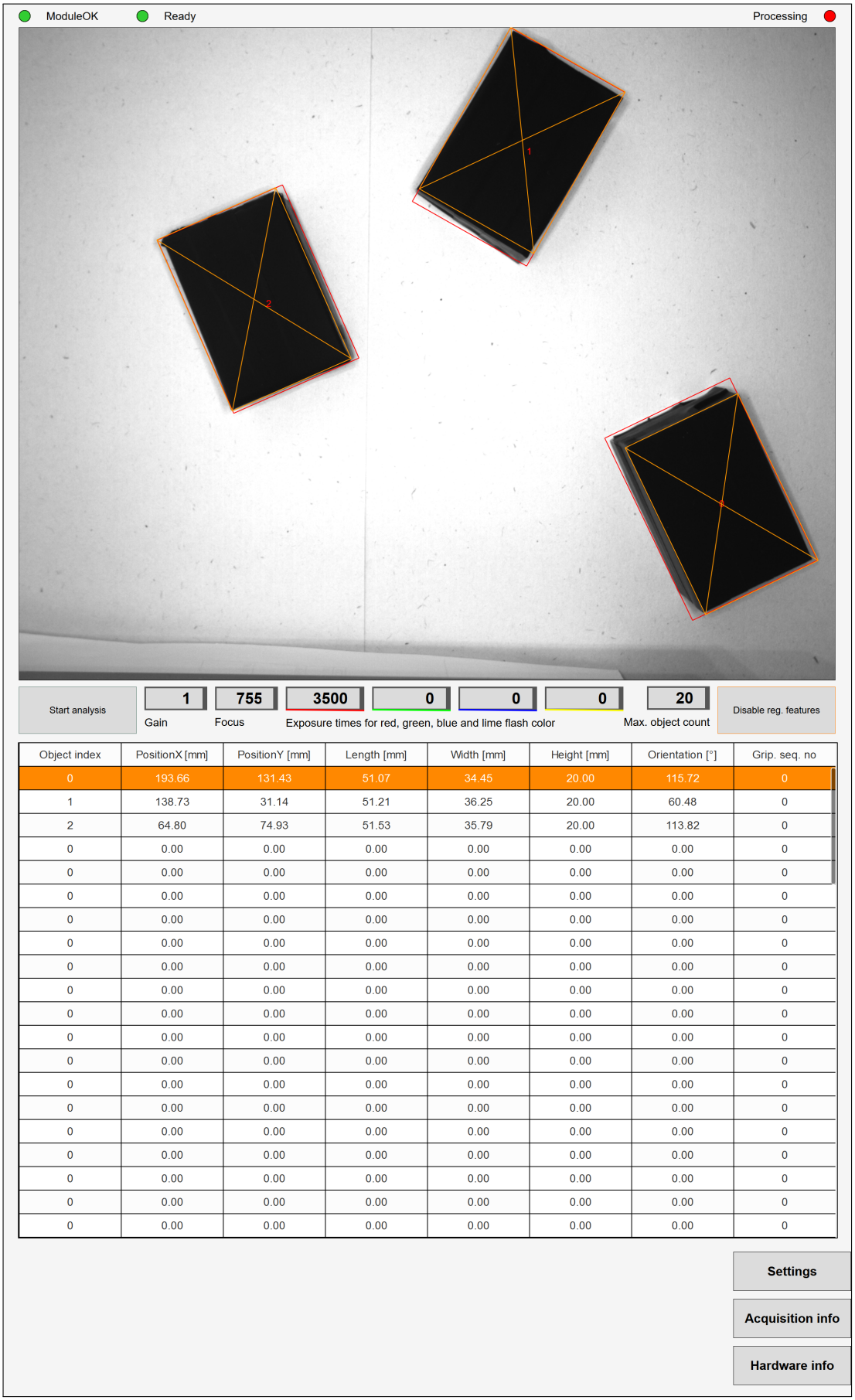
Z pohledu komunikace mezi uživatelem a aplikací hraje vizualizace klíčovou roli. Představuje pro něho hlavní zdroj informací a umožňuje mu provádět do běhu aplikace zásahy. Měla by mu poskytovat všechna potřebná data, ale ne na úkor přehlednosti. Zároveň by měla umožňovat jednoduché a intuitivní ovládání všech důležitých součástí zařízení. Uživateli by měla být práce s vizualizací příjemná a měla by mu umožnit efektivně a správně interpretovat aktuální stav prováděných operací. To jsou základní vlastnosti vizualizace vyplývající ze zdrojů [20] a [21].

Pro implementaci uživatelské vizualizace bylo zvoleno prostředí mapp View založené na webových technologiích, a to proto, že se jedná o aktuální a hojně používaný vizualizační prostředek od B&R. Uživateli je hotová vizualizace dostupná pomocí jakéhokoliv běžného internetového prohlížeče. Vlastní proces tvorba vizualizace spočívá ve využití integrovaných vizualizačních modulů, tzv. *widgetů*, jichž je široká škála – od jednoduchých tlačítek až po složité widgety například pro vykreslování grafů nebo SVG elementů.

Již vícekrát bylo zmíněno, že obsah této práce je pouze částí daleko většího projektu. Součástí tohoto projektu je i vizualizace, v níž byla strojovému vidění vyhrazena jedna strana. Vzhledem k tomu, že se jedná o vizualizaci s velkým rozlišením 1024x12300 pixelů, představuje jedna strana dostatečný prostor. Jak ve výsledku tato strana vypadá, je možné pozorovat na obrázku 4.1. V následujícím texu budou postupně popsány všechny prvky, které se na ní nachází, v pořadí odshora dolů.

Prvně se na straně nachází tři indikátory, které vyjádřují binární informace o aktuálním stavu Smart Sensoru. První z nich, *ModuleOK*, vypovídá o tom, zda je Smart Sensor připojen a úspěšně komunikuje, další (*Ready*) o tom, zda je připraven k použití – může být zahájeno pořízení a zpracování snímků, a poslední (*Processing*) o tom, zda aktuálně pořizování nebo zpracování snímku probíhá.

Pod těmito indikátory následuje snímek, poslední, jenž byl pořízen. Z hlediska práce s jakýmkoliv kamerovým systémem je zobrazení snímku velmi podstatné, proto je mu také v rámci strany vizualizace vyhrazena velká část. Poslední pořízený snímek je vždy dostupný na IP adrese Smart Sensoru, který se z tohoto hlediska chová jako server, proto může být jednoduše zobrazen widgetem *WebViewer* (webový prohlížeč). Ve skutečnosti byl však zvolen trochu odlišný přístup. WebViewer zobrazí snímek vždy větší, než je velikost widgetu a snímek tak není zobrazen celý naráz. To vytváří nežádoucí situaci, ve které je uživatel pro zobrazení celého snímku nucen použít rolovacích lišt. Proto je v PLC v uživatelském oddílu paměti (*user partition*) vytvořen html soubor, který snímek načte a zobrazí ve zvolené velikosti. Více o tomto souboru bylo uvedeno v části 3.3. Widget WebViewer pak zobrazuje tento html soubor a tak je možné zobrazit celý snímek v takřka libovolné velikosti.



Object index	PositionX [mm]	PositionY [mm]	Length [mm]	Width [mm]	Height [mm]	Orientation [°]	Grip. seq. no
0	193.66	131.43	51.07	34.45	20.00	115.72	0
1	138.73	31.14	51.21	36.25	20.00	60.48	0
2	64.80	74.93	51.53	35.79	20.00	113.82	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0
0	0.00	0.00	0.00	0.00	0.00	0.00	0

Obr. 4.1: Vizualizace – celá strana

Snímek je překryt dvěma *Paper* widgety, které slouží k zobrazení SVG elementů. První z nich vykresluje SVG řetězec s nalezenými objekty před korekcí, druhý po korekci. Díky tomu jsou uživateli výsledky zpracování snímku podány velmi srozumitelným způsobem, který mu umožní jejich rychlou kontrolu.

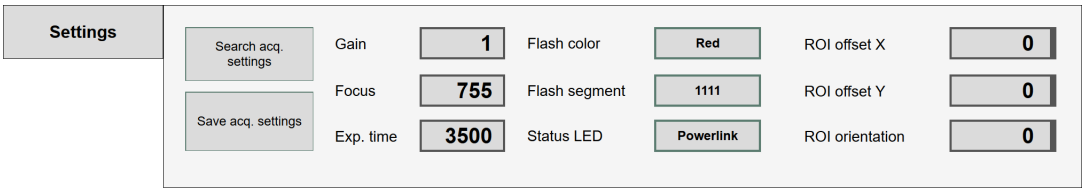
Přímo pod snímkem se nachází nejčastěji používané ovládací prvky. Je to v první řadě tlačítko k zahájení Blob analýzy. Co zmáčknutí tohoto tlačítka způsobí, bylo důkladně popsáno v části 3.2. Vedle tlačítka jsou číselné vstupy pro základní nastavení Smart Sensoru. První z nich označený *Gain* nastavuje zesílení jasu pořízeného snímku. Hodnota větší než jedna může být použita v nevyhovujících světelných podmínkách, nutno však poznamenat, že spolu se zesílením jasu snímku dochází též k zesílení šumu. Pokud je to možné, je doporučeno se nastavení většímu jedné vyhnout. Vedle je číselný vstup označený *Focus* pro nastavení zaostření objektivu – od minimální vzdálenosti (hodnota 2200) až do nekonečna (hodnota 0). Následují čtyři číselné vstupy stanovující časy expozic při nasvícení červeným, zeleným, modrým a žlutým světlem. Mechanismus vícenásobné expozice byl opět podrobně popsán v části 3.2. Posledním vstupem lze omezit maximální počet výsledků, který může při jednom zpracování snímku vrátit Blob Vision funkce (maximální počet je 20). Úplně napravo pak je tlačítko sloužící k vypnutí/zapnutí uvažování tzv. *region features* při vyhodnocení snímku Blob Vision funkcí (příznaky *Rectangularity*, *Circularity* a *Anisometry* popsané v části 1.3.1).

Pod těmito ovládacími prvky se nachází velká tabulka, v níž jsou vypsány všechny dostupné informace o výsledcích analýzy snímku. Jedná se vlastně o obsah pole struktur typu `VITemplateObjectType` po proběhnutí celého řetězce operací popsaných v části 3.2. V prvním sloupci jsou vypsány indexy do tohoto pole, v dalších dvou jsou pozice objektu ve směru os x a y (*PositionX* a *PositionY*), dále je tu sloupec s délkou (*Length*) a šířkou (*Width*) objektu. Následující sloupec s údaji o výšce objektů (*Height*) lze využít též jako vstup a simulovat tím roli nadřazeného systému. V dalším sloupci (*Orientation*) je uvedeno natočení objektu a konečně, v posledním sloupci označeném *Grip. seq. no* (*gripping sequence number*), jsou pořadová čísla odebrání objektů.

4.1 Boční panel *Settings*

První z bočních panelů obsahuje ovládací prvky týkající se nastavení Smart Sensoru. V první řadě jsou to dvě tlačítka *Search acq. settings* a *Save acq. settings*. První z nich je namapované na proměnnou `Cmd.SearchAcquisitionSettings` a spouští automatické hledání nastavení expozice s barvou nasvícení nastavenou výběrem z rozbalovací nabídky *Flash color* a konfigurací LED segmentů zvolou z rozbalovací nabídky *Flash segment*. Nalezené nastavení expozice je zobrazeno v polích

s číselnými výstupy *Gain*, *Focus* a *Exp. time* a lze jej potvrdit stiskem tlačítka *Save acq. settings* nebo odmítnout opakovaným stiskem prvního tlačítka, jehož text je v tu chvíli změněn na *Discard acq. settings*.



Obr. 4.2: Vizualizace – boční panel *Settings*

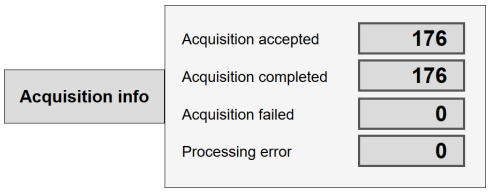
Vedle těchto tlačítek jsou pole s číselnými výstupy (*Gain*, *Focus* a *Exp. time*), které nesou informaci o aktuálních hodnotách nastavení expozice. Tyto číselné výstupy jsou v daném pořadí namapovány na proměnné `Status.ReadGainLevel`, `Status.ReadFocus` a `Status.ReadExposureTime`.

Dále napravo se nachází tři rozbalovací nabídky. První z nich (*Flash color*) je namapována na proměnnou `Cfg.FlashColor01` a lze pomocí ní nastavit barvu nasvícení, která bude použita při hledání nastavení expozice. Výběr je následující: *Off, Red, Green, Blue* a *Lime*. Další rozbalovací nabídka (*Flash segment*) umožňuje volit, jaké LED segmenty budou pro nasvícení použity, je mapována na proměnnou `Cfg.FlashSegment01`. Nabídka obsahuje čísla 0 až 15 v binárním zápisu (*0000* až *1111*). Každé ze čtveřice čísel nula nebo jedna znamená zapnuto nebo vypnuto pro daný LED segment. První zprava nastavuje použití prvního LED segmentu, druhé číslo zprava segment druhý atd. Např. při volbě *0101* bude k nasvícení použit pouze první a třetí segment. Poslední rozbalovací nabídkou je *Status LED*, kterou je prováděna volba barvy světla statusových LED (mapována na proměnnou `Cfg.StatusLED`). První možností v této nabídce je *Powerlink*, při které statusové LED pomocí červené a zelené barvy zobrazují stav komunikace přes sběrnici Powerlink. Krom této možnosti je zde také možnost *Off, Green, Red* a *Blue*.

Poslední tři ovládací prvky se týkají nastavení ROI. V současné době je ve Smart Sensoru implementována změna oblasti zájmu pomocí mapovaných proměnných pouze v možnosti posunutí jejího počátku – vstupní číselná pole *ROI offset X* a *ROI offset Y* mapované na proměnnou `Cfg.OffsetROIX` a `Cfg.OffsetROIY`. Proměnná pro orientaci oblasti zájmu `Cfg.OffsetROIOrientation` není v aplikaci používána, proto není pole *ROI orientation* aktivní.

4.2 Boční panel *Acquisition info*

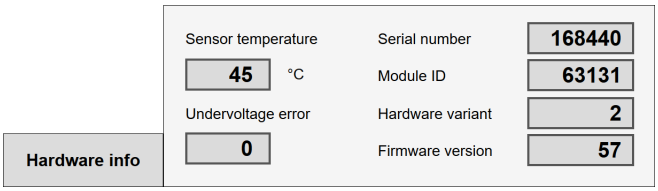
Tlačítkem *Acquisition info* je vysunut boční panel zobrazující údaje týkající se pořízení snímku. První tři údaje mají statistický charakter; je to počítadlo přijatých příkazů k pořízení snímku (namapováno na proměnnou `Status.AcceptedAcquisitionCnt`), počítadlo úspěšně pořízených snímků (`Status.CompletedAcquisitionCnt`) a počítadlo snímků, jejichž pořízení selhalo (`Status.FailedAcquisitionCnt`). Čtvrtý údaj specifikuje chybový status kamery (`Status.ImageProcessingError`).



Obr. 4.3: Vizualizace – boční panel *Acquisition info*

4.3 Boční panel *Hardware info*

Boční panel *Hardware info* obsahuje informace o Smart Sensoru z hlediska HW. Pomocí číselných výstupů je zobrazena aktuální teplota Smart Sensoru (namapováno na proměnnou `HwInfo.SensorTemperature`), příznak siganlizující chybu napájení (`HwInfo.UndervoltageError`), výrobní číslo (`HwInfo.SerialNumber`), identifikační číslo (`HwInfo.ModuleID`), varianta hardware (`HwInfo.HardwareVariant`) a verze firmware (`HwInfo.FirmwareVersion`).



Obr. 4.4: Boční panel *Hardware info*

5 Závěr

Při zpracování rešerše týkající se použití strojového vidění v průmyslu z hlediska HW byl zmapován aktuální trh a nalezená dostupná řešení rozdělena do základních tří kategorií podle stupně integrace do řídicích systémů. Rešerše z hlediska SW byla zaměřena na konkrétní algoritmy pro zpracování obrazu za účelem identifikace objektů implementované ve Smart Sensoru. Je to Blob analýza založená na modelech (*Model-based Blob Analysis*) a Matching.

V druhé kapitole obsahující navrženou koncepci řešení byl proveden popis scény snímání a ověřeno, že má Smart Sensor v dostupné konfiguraci pro snímání vyhovující vlastnosti. Dále byl proveden rozbor, na jehož základě byla Blob analýza založená na modelech pro danou aplikaci vyhodnocena jako vhodnější algoritmus, než-li Matching. Velká pozornost byla věnována zpracování dat získaných Blob analýzou, a to jmenovitě kalibraci kamery (stanovení závislosti mezi pixely a milimetry), korekci zkreslení daného perspektivou snímání (odvození vztahů pro opravu rozměrů a pozic obrobků) a detekci kolize chapadla robota při odebírání obrobků (včetně stanovení pořadí odebírání).

V další části diplomové práce byla popsána realizace programového řešení. Veškeré navržené operace s daty získanými Blob analýzou (korekce zkreslení, detekce kolize atd.) byly implementovány jako funkce jedné knihovny nazvané „Vision Template“ (VITemplate). Při jejím vývoji byl kladen důraz na vysokou míru univerzálnosti, díky čemuž je řešení do značné míry modulární a může být použito jako šablona pro Vision část dalších projektů podobného rázu. Součástí knihovny jsou také základní struktury pro práci se Smart Sensorem a funkce určené pro vizualizaci výsledků v podobě SVG. Postupným voláním jednotlivých funkcí knihovny VITemplate je možno provést operace s daty získanými Smart Sensorem nutné k tomu, aby byly získány souřadnice nalezených obrobků použitelné k řízení robota. Použití knihovny je demonstrováno v programovém modulu ViMain.

Na závěr byla vytvořena uživatelská vizualizace ve vývojovém prostředí Mapp View založeném na webových technologiích. Vizualizace je zobrazitelná běžně dostupnými webovými prohlížeči a lze skrze ni nastavovat Smart Sensor, řídit zpracování obrazu a sledovat výsledky Blob analýzy.

Všechny dílčí části diplomové práce byly otestovány na reálném systému. Knihovna VITemplate s volbou kalibrace kamery měřením zorného pole byla úspěšně otestována i s robotickým ramenem na reálném průmyslovém pracovišti.

Literatura

[1] B&R. Vision systems. br-automation.com [online]. ©2020 [cit. 2020-05-24]. Dostupné z: <https://www.br-automation.com/en/products/vision-systems/>.

[2] STEGER, C.; ULRICH, M.; WIEDERMANN, CH. Machine Vision Algorithms and Applications, 2nd, completely revised and enlarged Edition. John Wiley & Sons. 2018. ISBN 3527413650, 9783527413652.

[3] BAUER, N.; ALBRECHT, N. Guideline for Industrial Image Processing. Fraunhofer-Allianz Vision, 2003. ISBN 3816762964, 9783816762966.

[4] AUTOMATED IMAGING ASSOCIATION (AIA). Computer Vision vs. Machine Vision. visiononline.org [online]. ©2009-2020 [cit. 2019-05-24]. Dostupné z: <https://www.visiononline.org/vision-resources-details.cfm/vision-resources/Computer-Vision-vs-Machine-Vision/content_id/4585>.

[5] VIDOLAB. Machine Vision vs Computer Vision: What's the Difference? computer-vision-ai.com [online]. [cit. 2019-05-24]. Dostupné z: <https://computer-vision-ai.com/blog/machine-vision-vs-computer-vision/>.

[6] COGNEX CORPORATION. Introduction to Machine Vision: A guide to automating process & quality improvements. cognex.com [online]. ©2020 [cit. 2020-05-24]. Dostupné z: <https://www.cognex.com/what-is/machine-vision>.

[7] BASLER. Basler ace Series. baslerweb.com [online]. ©2020 Basler AG [cit. 2020-05-24]. Dostupné z: <https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/#productline=aceclassic>.

[8] BASLER. Basler Lenses 1/2.5"and Basler Lenses 2/3". baslerweb.com [online]. ©2020 Basler AG [cit. 2020-05-24]. Dostupné z: <https://www.baslerweb.com/en/products/vision-components/basler-lenses/>.

[9] VISION SYSTEMS DESIGN. Basler releases new machine vision intelligent lighting solutions. vision-systems.com [online]. ©2020 Endeavor Business Media, LLC. [cit. 2020-05-24]. Dostupné z: <https://www.vision-systems.com/cameras-accessories/article/16748299/basler-releases-new-machine-vision-intelligent-lighting-solutions>.

[10] BASLER. Frame Grabbers: microEnable 5 marathon. baslerweb.com [online]. ©2020 Basler AG [cit. 2020-05-24]. Dostupné z: <https://www.baslerweb.com/en/products/vision-components/frame-grabbers/microenable-5-marathon/>.

[11] ALLEN-BRADLEY BY ROCKWELL AUTOMATION. MultiSight Photoelectric Vision Sensors. ab.rockwellautomation.com [online]. ©2020 Rockwell Automation, Inc. [cit. 2020-05-24]. Dostupné z: <https://ab.rockwellautomation.com/sensors-switches/photoelectric-vision-sensors/48ms-multisight-photoelectric-sensors>.

[12] BECKHOFF. TwinCAT Vision: Integrating machine vision into automation. beckhoff.com [online]. Beckhoff Automation GmbH & Co. KG [cit. 2020-05-24]. Dostupné z: <https://download.beckhoff.com/download/document/catalog/Beckhoff_TwinCAT_Vision_e.pdf>.

[13] OMRON INDUSTRIAL AUTOMATION. Smart Camera FHV7 Series. ia.omron.com [online]. ©2007-2020 OMRON Corporation [cit. 2020-05-24]. Dostupné z: <http://www.ia.omron.com/products/family/3740/download/catalog.html>.

[14] ROSIN, L. PAUL. Measuring Shape: Ellipticity, Rectangularity, and Triangularity. Machine Vision and Applications, Volume 14, Issue 3, strany 172–184. 2003.

[15] MVTEC HALCON. HALCON Operator Reference. Version 12.0.2. mvtec.com [online]. ©2020 MVTec Software GmbH [cit. 2020-05-24]. Dostupné z: <https://www.mvtec.com/doc/halcon/12/en/index.html>.

[16] MVTEC HALCON. Solution Guide II-B Shape-Based Matching. mvtec.com [online]. ©2020 MVTec Software GmbH [cit. 2020-05-24]. Dostupné z: <http://download.mvtec.com/halcon-9.0-solution-guide-ii-b-shape-based-matching.pdf>.

[17] MVTEC HALCON. Shape-based matching with MVTec HALCON: speedup vs. robustness, advanced parameters. mvtec.com [online]. ©2020 MVTec Software GmbH [cit. 2020-05-24]. Dostupné z: <https://www.mvtec.com/news-press/video/detail/shape-based-matching-with-mvtec-halcon-speedup-vs-robustness-advanced-parameters>.

[18] EDMUND OPTICS. Imaging Resource Guide: Understanding Focal Length and Field of View. edmundoptics.com [online]. ©2020 Edmund Optics Inc. [cit. 2020-05-24]. Dostupné z: <https://www.edmundoptics.com/knowledge-center/industry-expertise/imaging-optics/imaging-resource-guide/>.

[19] W3C – WORLD WIDE WEB CONSORTIUM. Scalable Vector Graphics (SVG) 1.1 (Second Edition). w3.org [online]. ©2011 W3C [cit. 2020-05-24]. Dostupné z: <https://www.w3.org/TR/SVG11/>.

[20] ASEE – AMERICAN SOCIETY FOR ENGINEERING EDUCATION. HMI Design: An Analysis of a Good Display for Seamless Integration between User Understanding and Automatic Controls, ©ASEE 2020 [cit. 2020-05-24]. Dostupné z: <https://peer.asee.org/ hmi-design-an-analysis-of-a-good-display-for-seamless-integration-between-user-understanding-and-automatic-controls>.

[21] B&R. Introduction to visualization. V. 1.0.3.1, ©B&R 2015.

Seznam symbolů, veličin a zkratek

AFOV	úhlové zorné pole (*angular field of view*)
APC	Automation PC
CV	počítačové vidění (*computer vision*)
FOV	zorné pole (*field of view*)
FPGA	programovatelné hradlové pole (*field-programmable gate array*)
HMI	uživatelské rozhraní (*human-machine interface*)
HW	hardware
IR	infračervené (*infrared*)
LED	světlo vyzařující dioda (*light-emitting diode*)
MV	strojové vidění (*machine vision*)
NCC	normalizovaná vzájemná korelace (*normalized cross-correlation*)
OCR	rozpoznání textu (*optical character recognition*)
PLC	programovatelný logický automat (*programable logic controller*)
ROI	oblast zájmu (*region of interest*)
SAT	teorém o dělicí ose (*separating axis theorem*)
SW	software
UV	ultrafialové (*ultra violet*)

Seznam příloh

A Obsah přiloženého CD

Na přiloženém CD se krom elektronické verze práce (včetně zdrojových souborů pro generování pomocí programu LaTeX) nachází další obsah.

Složka `Zdrojove_soubory_VITemplate` obsahuje zdrojové soubory knihovny VITemplate. V dalších dvou složkách (`Zdrojove_soubory_ViMain` a `Zdrojove_soubory_ViImLoader`) jsou umístěny zdrojové soubory příslušných programových modulů. Samotný projekt s programovým řešením pro PLC je zkomprimován do archivu, který se nazývá `VITemplate_project.zip`. Projekt byl vyvíjen a testován v Automation Studiu verze 4.7.3.93 SP. Posledním souborem nazvaným `Zaznam_z_testovani.mkv` je videozáznam z testování systému na reálném pracovišti s robotickým ramenem. Videozáznam je sice horší kvality a jedná se jednu z prvních verzí programového řešení, přesto ale dává čtenáři alespoň rámcovou představu o využití této práce.

A.1 Strukturovaný výpis souborů a složek

- `/` ..... kořenová složka přiloženého CD
  - `Zdrojove_soubory_ViImLoader` ..... zdr. soubory prog. modulu ViImLoader
    - `ANSIC.prg` ..... pomocný soubor AS
    - `Main.c` ..... zdrojový kód
    - `Types.typ` ..... definice datových typů
    - `Variables.var` ..... definice proměnných
  - `Zdrojove_soubory_ViMain` ..... zdr. soubory prog. modulu ViMain
    - `ANSIC.prg` ..... pomocný soubor AS
    - `Main.c` ..... zdrojový kód
    - `Types.typ` ..... definice datových typů
    - `Variables.var` ..... definice proměnných
  - `Zdrojove_soubory_VITemplate` ..... zdr. soubory knihovny VITemplate
    - `ANSIC.lby` ..... pomocný soubor AS
    - `VITemplate.fun` ..... definice rozhraní funkcí
    - `VITemplate.typ` ..... definice datových typů
    - `VITemplate.var` ..... konstanty knihovny
    - `VITemplateCorrPersp.c` ..... zdrojový kód
    - `VITemplateDrawSvgObj.c` ..... zdrojový kód
    - `VITemplateDrawSvgRoi.c` ..... zdrojový kód
    - `VITemplateGetBlobData.c` ..... zdrojový kód
    - `VITemplateGripSeqNum.c` ..... zdrojový kód
    - `VITemplateRemColl.c` ..... zdrojový kód
  - `DP_Vostrel_Tomas_2020.pdf` ..... elektronická verze DP
  - `VITemplate_project.zip` ..... archiv s projektem pro AS
  - `Zaznam_z_testovani.mkv` ..... videozáznam z testování na pracovišti s robotem